Project 1

Building a CI/CD Pipeline for a Retail Company

First of all, thanks very much to edureka and instructors for giving me confidence that I can do any type of devops projects. I just loved working on this project. I have chosen project 1 out of two projects. So I implemented the 1st project.

github respository: https://github.com/bejoykoottumkal/devops-lab.git

Business Challenge/Requirement:

ABC Technologies is a leading online retail store, and it has recently acquired a large retail offline business store. The business store has a large number of stores across the globe but is following the conventional pattern of development and deployment. As a result, it has landed at a great loss and is facing the following challenges.

- Low available
- Low scalable
- Low performance
- Hard to built and maintain
- Developing and deploying are time-consuming

ABC will acquire the data from all these storage systems and plans to use it for analytics and prediction of the firm's growth and sales prospects. In the first phase, ABC has to create the servlets to add a product and display product details. Add servlet dependencies required to compile the servlets. Create an HTML page that will be used to add a product. The team is using Git to keep all the source code.

ABC has decided to use the DevOps model. Once source code is available in GitHub, we need to integrate it with Jenkins and provide continuous build generation for continuous delivery as well as integrate with Ansible and Kubernetes for deployment. Use Docker Hub to pull and push images between Ansible and Kubernetes.

Problem Statements/Tasks:

We need to develop a CI/CD pipeline to automate the software development, testing, packaging, and deployment, reducing the time to market the app and ensuring good quality service is experienced by end users. In this project, we need to—

- Push the code to our GitHub repository.

- Create a continuous integration pipeline using Jenkins to compile, test, and package the code present in GitHub.
- Write Dockerfile to push the war file to the Tomcat server.
- Integrate Docker with Ansible and write the playbook.
- Deploy artifacts to the Kubernetes cluster
- Monitor resources using Prometheus.

Approach to Solve:

Task 1: Clone the project from the GitHub link shared in resources to your local machine. Build the code using Maven commands.

Task 2: Set up the Git repository and push the source code. Then, log in to Jenkins.

1. Create a build pipeline containing a job for each
    - One for compiling source code
    - Second for testing source code
    - Third for packing the code
2. Execute the CI/CD pipeline to execute the jobs created in step 1
3. Set up a master-slave node to distribute the tasks in the pipeline

Task 3: Write a Docket file. Create an Image and container on the Docker host. Integrate docker host with Jenkins. Create CI/CD job on Jenkins to build and deploy on a container.

1. Enhance the package job created in step 1 of task 2 to create a docker image.
2. In the Docker image, add code to move the war file to the Tomcat server and build the image.

Task 4: Integrate the Docker host with Ansible. Write an Ansible playbook to create an image and create a continuer. Integrate Ansible with Jenkins. Deploy Ansible-playbook. CI/CD job to build code on ansible and deploy it on docker container

- Deploy Artifacts on Kubernetes
- Write pod, service, and deployment manifest file
- Integrate Kubernetes with Ansible
- Ansible playbook to create deployment and service

Task 5: Using Prometheus, monitor the resources like CPU utilisation: Total Usage, Usage per core, usage breakdown, memory, and network on the instance by providing the endpoints on the local host. Install the node exporter and add the URL to the target in Prometheus.

Let's start…

Task 1:

Clone the project from the GitHub link shared in resources to your local machine. Build the code using Maven commands.

Approach i have followed:

Before setting up the CI/CD pipeline,

I checked and installed necessary software in my machine using power-shell and also installed Ubuntu console using the following codes.

1. Java:

Verified Java Installation:

java -version

2. Maven:

Verified Maven Installation:

mvn -version

3. Git:

Verified Git Installation:

git --version

4. Jenkins:

Verified Jenkins Installation:

Accessed Jenkins in a web browser at http://localhost:8080 (default) and follow the setup process.

5. Docker:

Verified Docker Installation:

docker –version

6. Kubernetes:

Verified Kubernetes Installation:

kube CTL

in AWS t3 Medium

7. Ansible:

Verify Ansible Installation:

ansible –version

in AWS t3 Medium

8. Prometheus:

Verified Prometheus Installation:

Access Prometheus in a web browser at http://localhost:9090 .

Additional Considerations :

- Ensured Service Availability
- Checked Jenkins, Docker, Ansible, Kubernetes , and Prometheus services are running and available.

Configuration Check:

- Checked and Verified the configurations for Jenkins, Docker, Ansible, Kubernetes and Prometheus.
- Network Ports
- Ensured that required ports are not blocked by firewalls. For example, Jenkins typically uses port 8080 and Prometheus uses port 9090.
- Documentation:
- Record Versions:
- Documented the versions of each installed software.
- As a given project is based on Java I have used Maven as a build tool to build the code. Maven and Java are installed already in the master machine. I have run the Maven compile, test, and package tasks successfully in the local environment and produced the results for each and also executed the Maven clean install command to build the code.

- Before this I have downloaded the sample Java code from the LMS into my own local environment and used git bash to upload to the below newly created public GitHub repository.

Repository: https://github.com/bejoykoottumkal/devops-lab.git

Step 1: Cloned the Project

Opened a terminal or command prompt on your local machine and ran the following command to clone the project from the GitHub repository:

git clone https://github.com/bejoykoottumkal/devops-lab.git

Step 2: Navigated to the Project Directory

Moved to the project directory using the cd command:

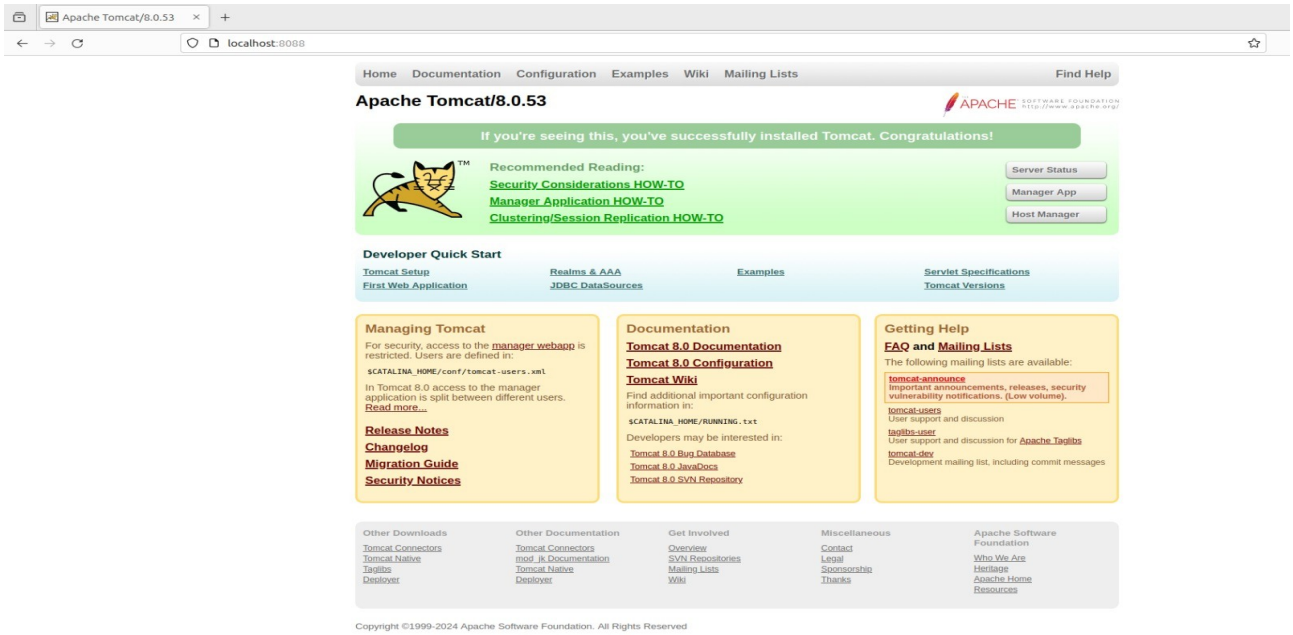
cd devops-lab

Step 3: Built the Code with Maven

Used Maven to build the project. Executed the following commands:

mvn clean
mvn compile
mvn test
mvn package

Step 4: Application Deployment using native tomcat installation for checking the whether application is up.



Copied the war file created in target folder to the webapps folder of tomcat and stared service. Application was available with url : http://localhost:8088/cicd/

References:

https://www.marcobehler.com/guides/mvn-clean-install-a-short-guide-to-maven

https://www.geeksforgeeks.org/maven-lifecycle-and-basic-maven-commands/

Task 2:

Set up the Git repository and push the source code. Then, log in to Jenkins.

1. Create a build pipeline containing a job for each

- **One for compiling source code**
- **Second for testing source code**
- **Third for packing the code**

2. Execute the CI/CD pipeline to execute the jobs created in step 1

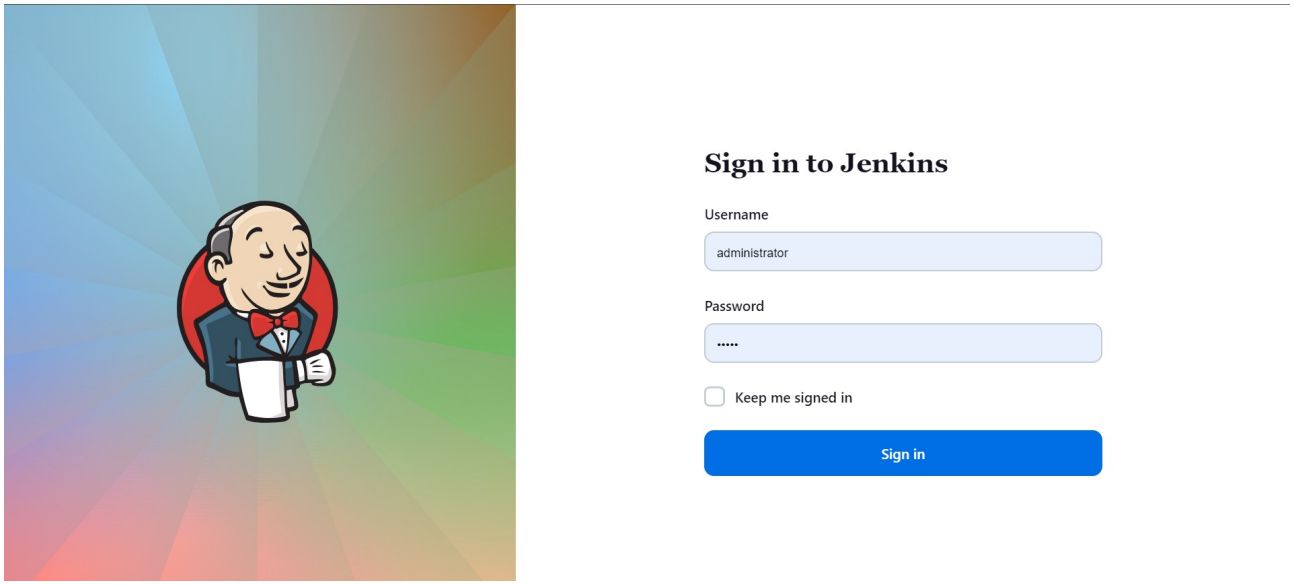
3. Set up a master-slave node to distribute the tasks in the pipeline

Approach i have followed:

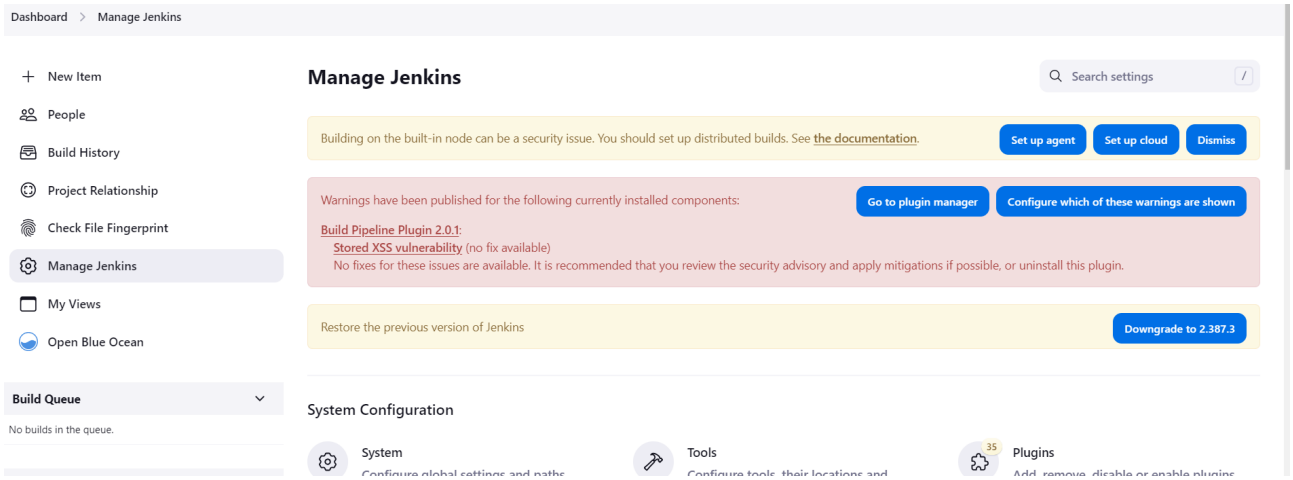
Step 1: Installed Jenkins

Downloaded and installed Jenkins on the machine, following the official instructions: https://www.jenkins.io/download/

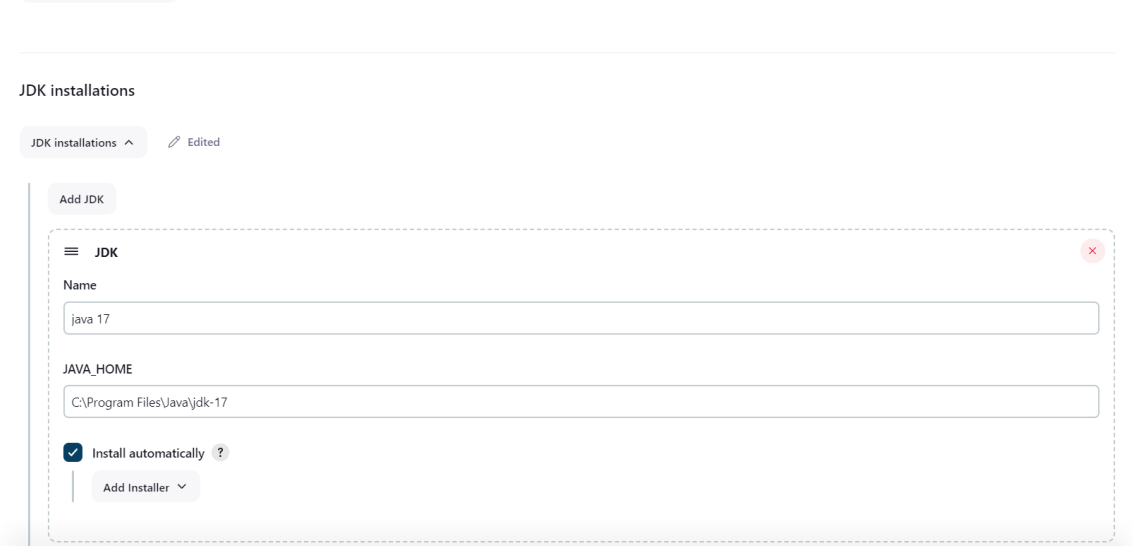
Started Jenkins after installation and log in



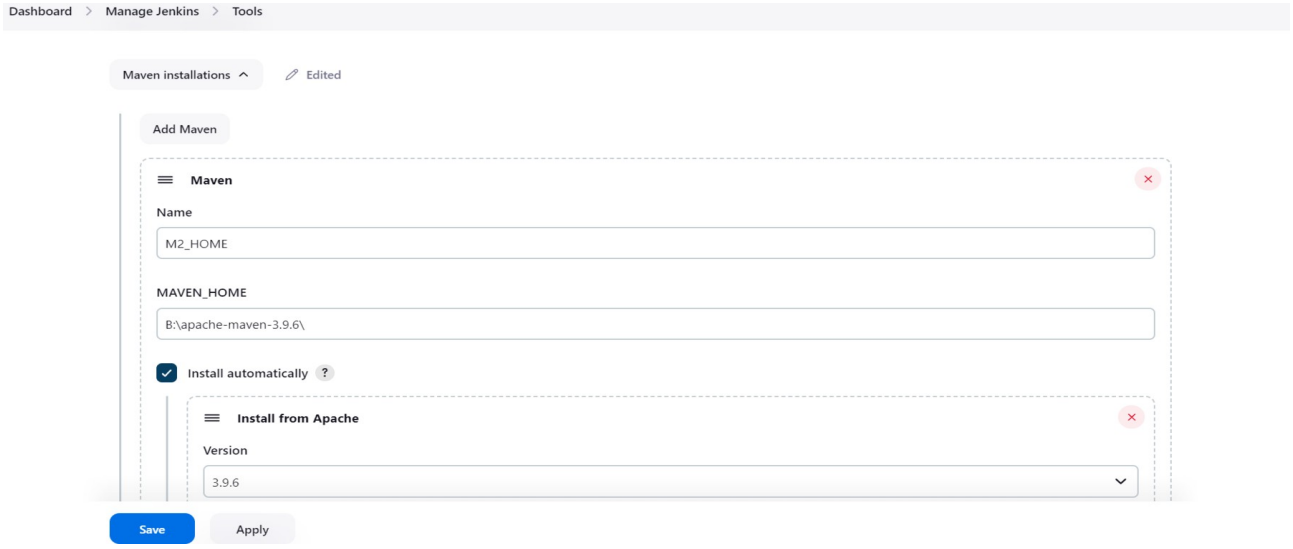
Global Tool Configurations.



Added JAVA HOME.



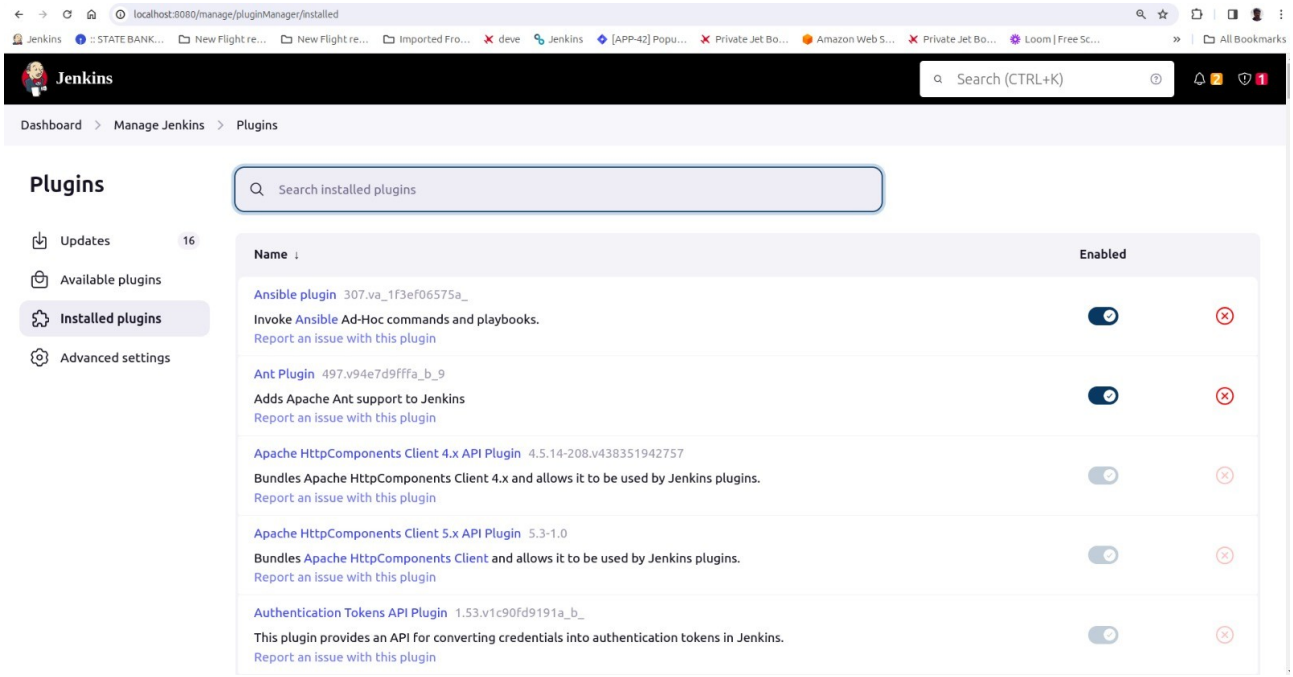
Added MAVEN HOME



Step 2: Created Jenkins Build Pipeline

Opened Jenkins in the web browser (http://localhost:8080).

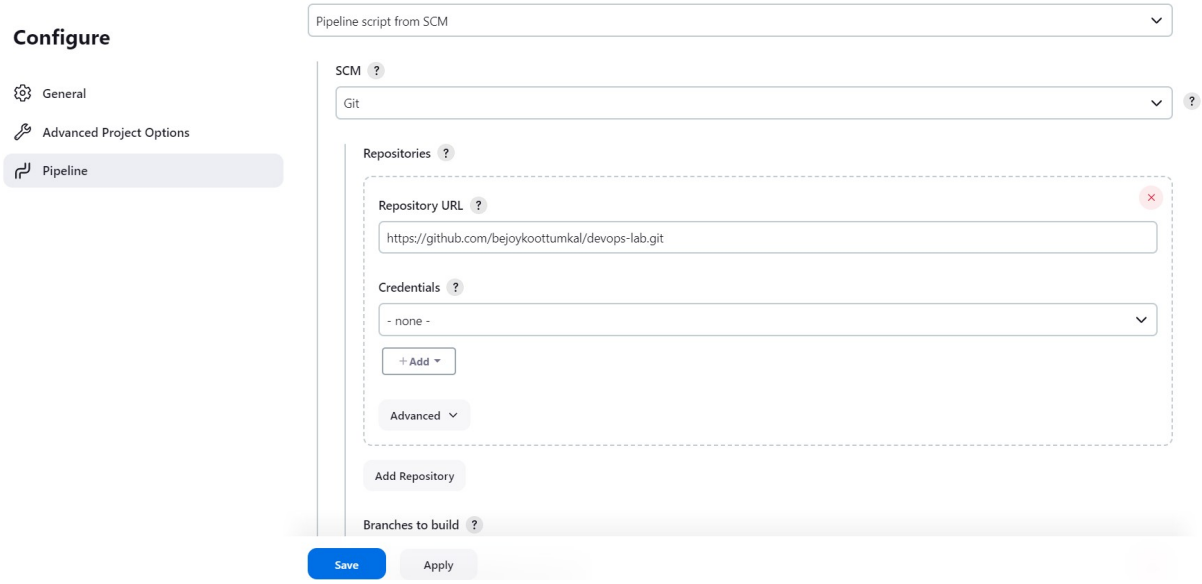
Installed the necessary plugins.



Step 3: Configured Jenkins Jobs

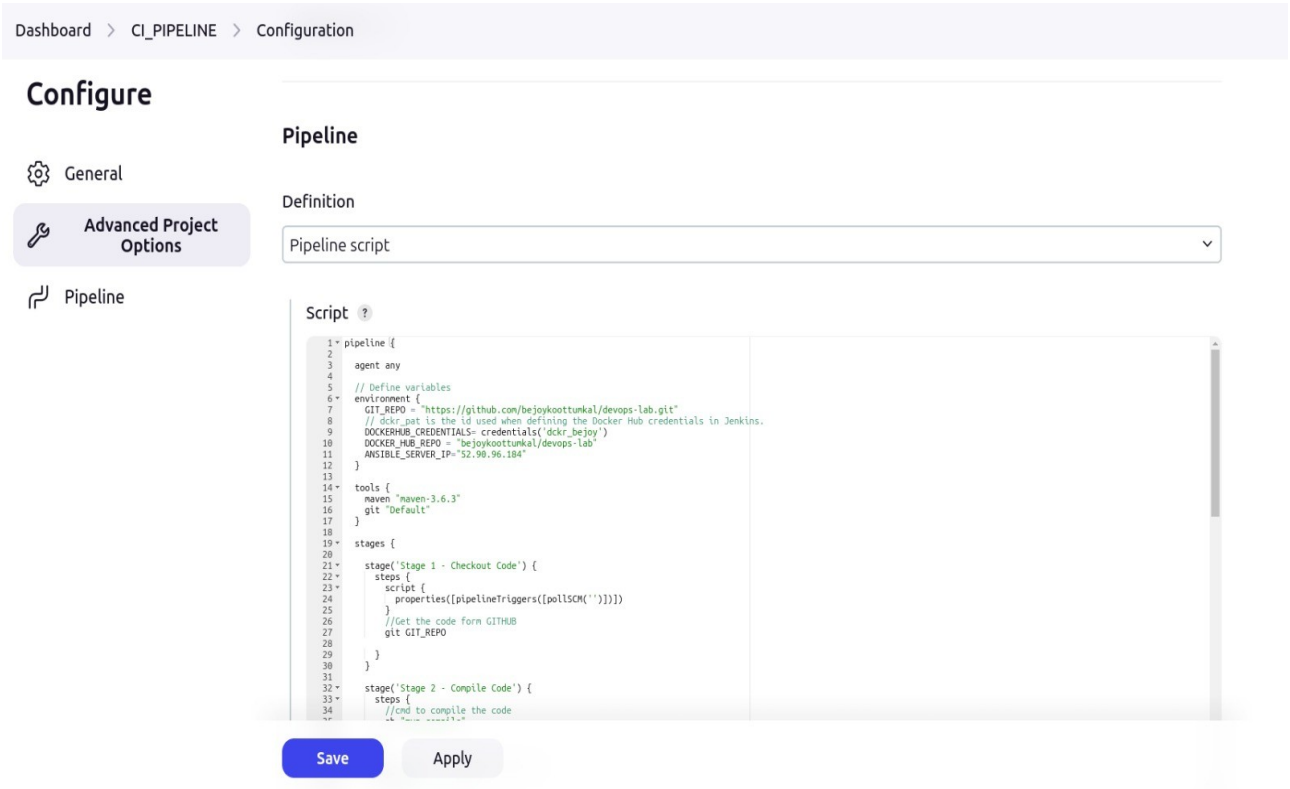
Created a new Jenkins job.

Configured it to pull the code from the Git repository.



Step 4: Created a Build Pipeline

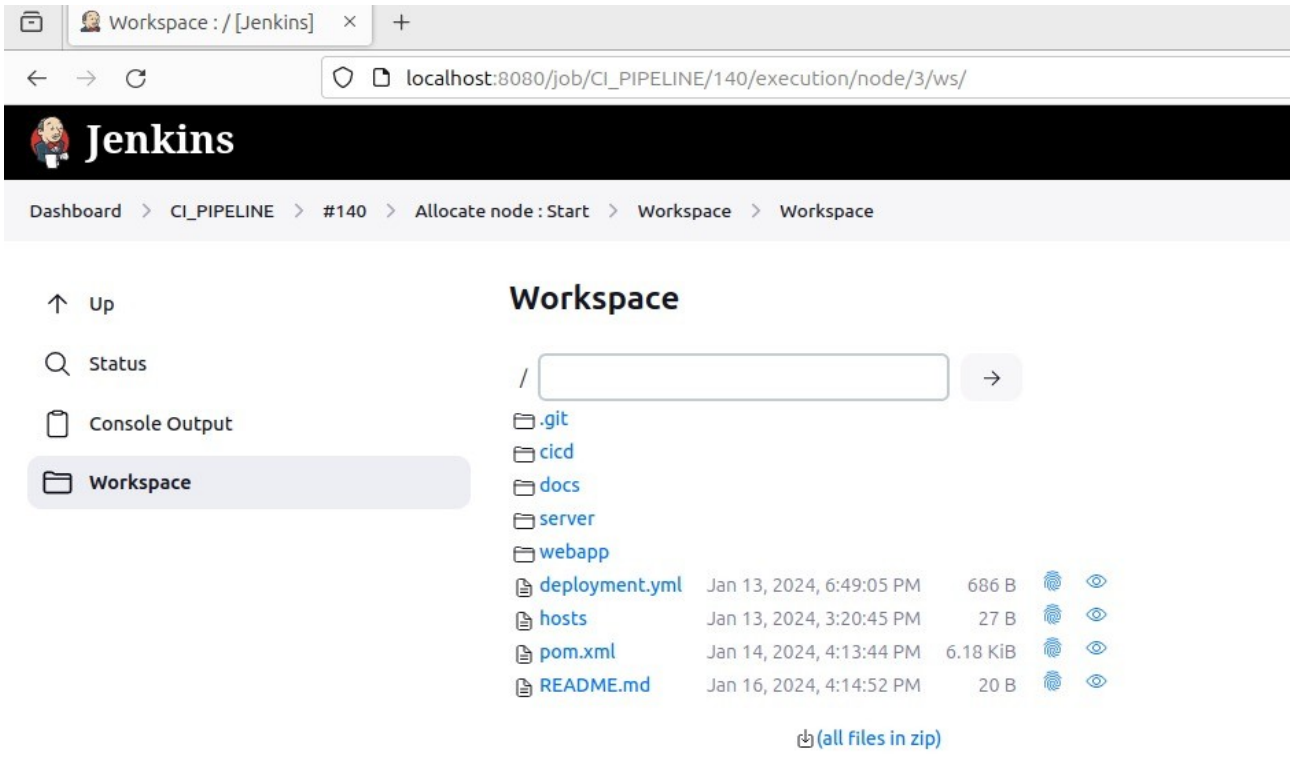
Created a new Jenkins job of type "Pipeline" added the pipeline script in it.



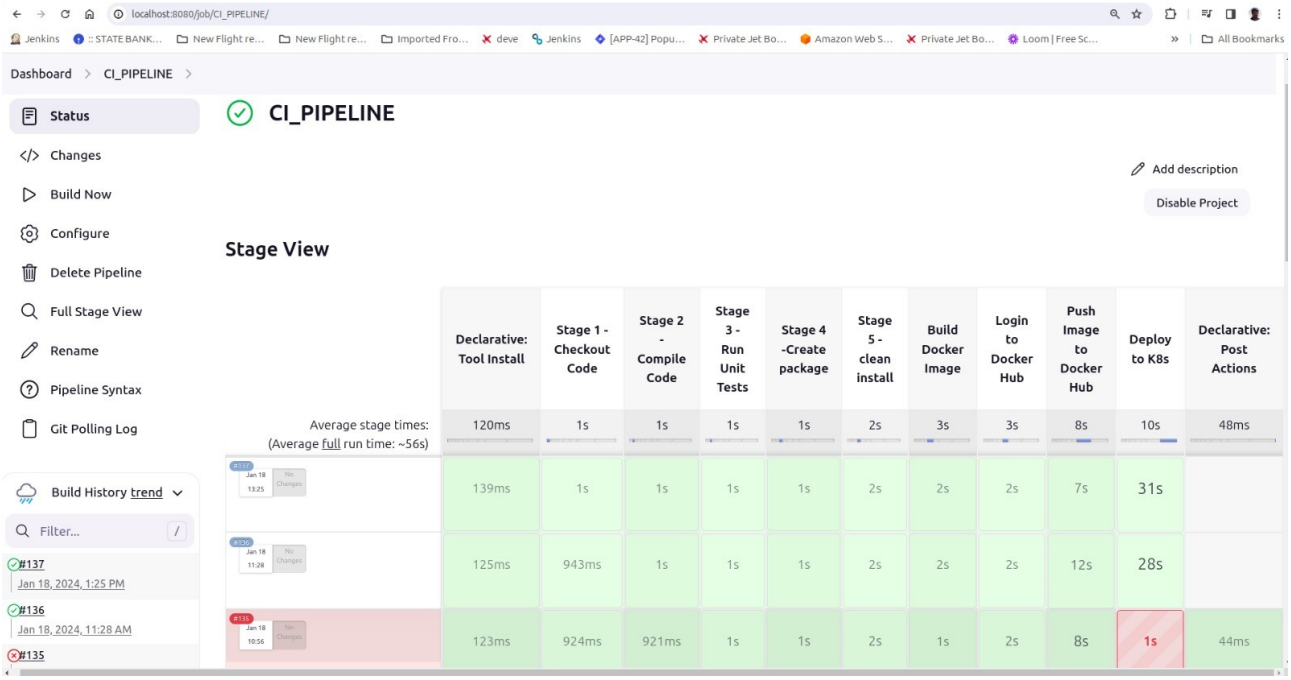
JenkinFile

```
39        stage('Stage 3 - Run Unit Tests') {
40          steps {
41            //cmd to run tests
42            sh "mvn test"
43          }
44        }
45
46        stage('Stage 4 -Create package') {
47          steps {
48            //cmd to create the build of project
49            sh "mvn package"
50          }
51        }
52
53        stage('Stage 5 - clean install') {
54          steps {
55            //cmd to create the build of project
56            sh "mvn clean install"
57          }
58        }
```

Target folder in Jenkins:

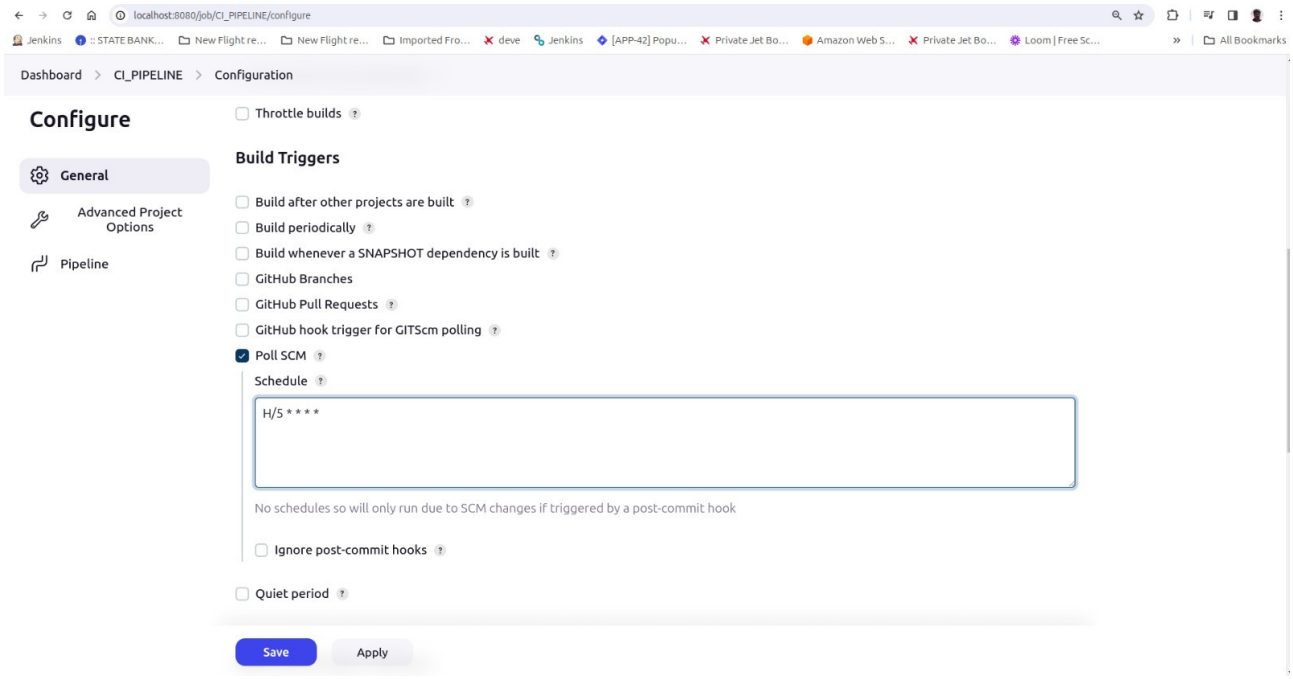


Execute the CI/CD pipeline to execute the jobs created in step 1 CI_PIPELINE:

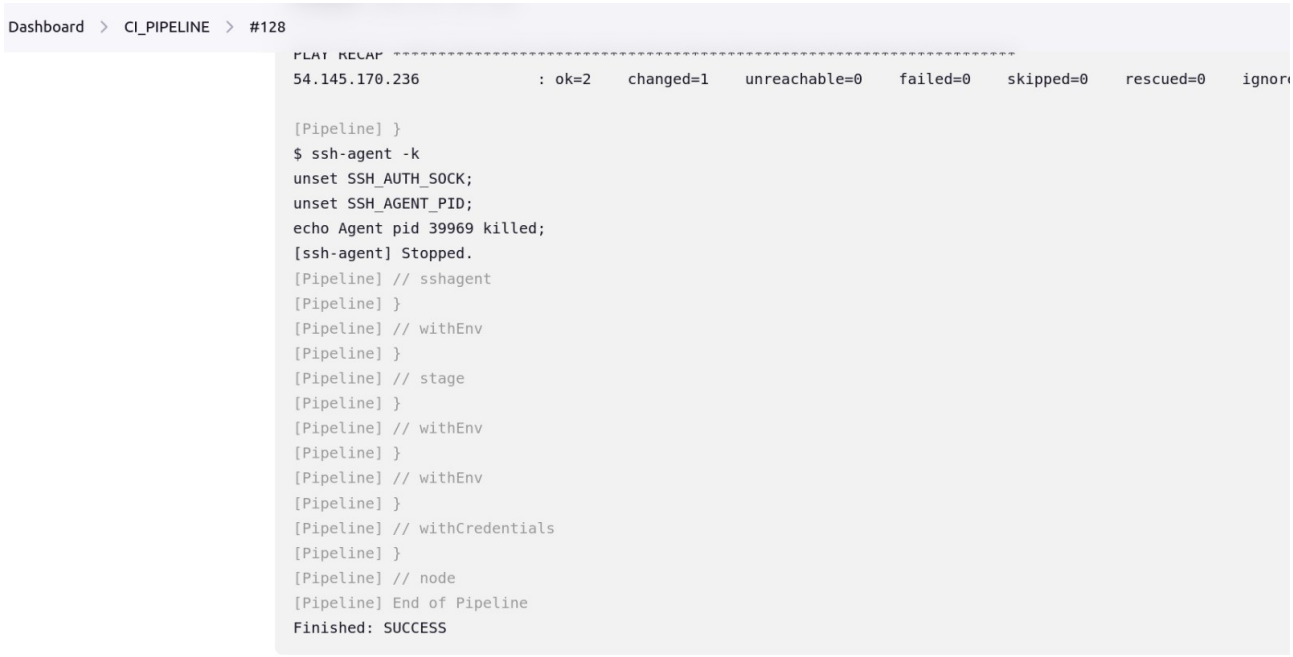


And I have configured the git polling as a cron job for a compile job that triggers for github code so that if whatever change happens then it will trigger and perform the given task.

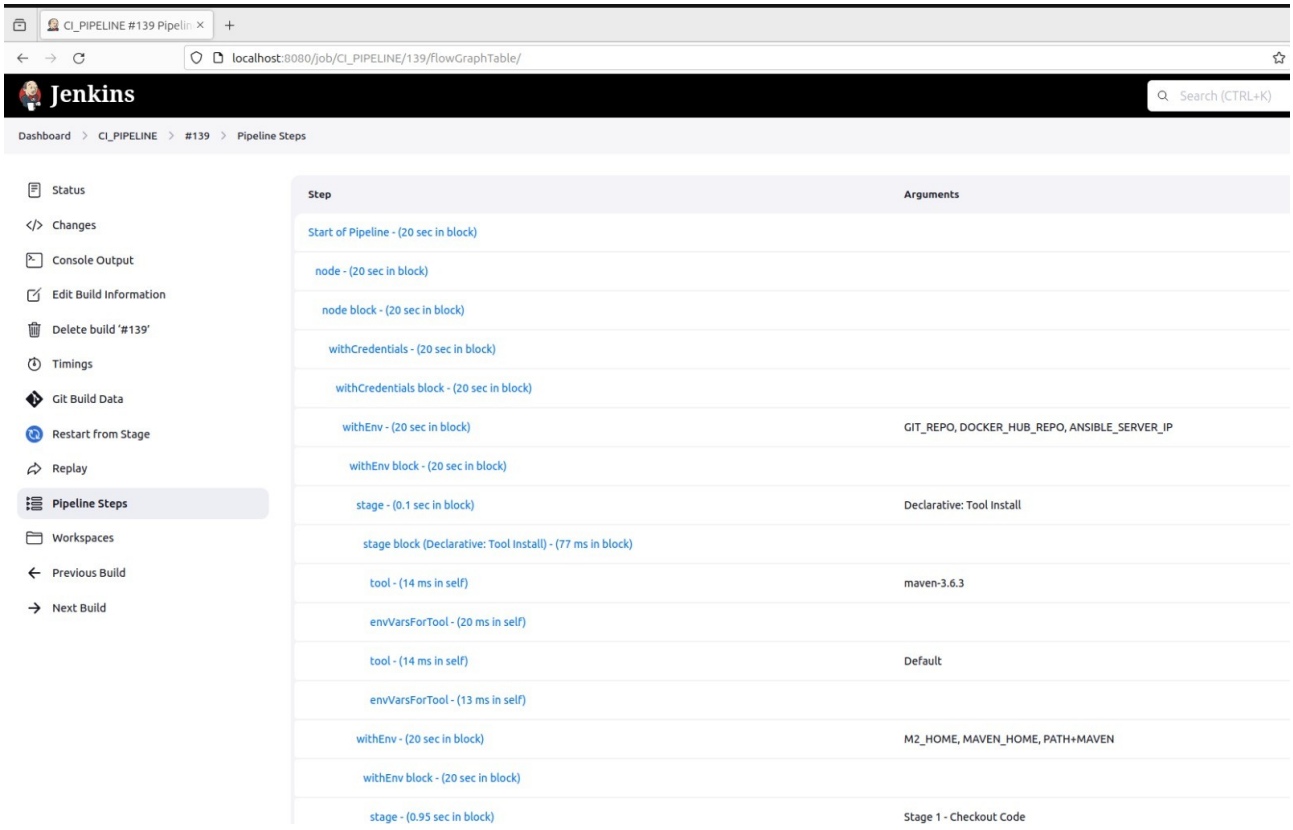
Poll SCM settings using cron job in compile job(triggers every 5 minutes):



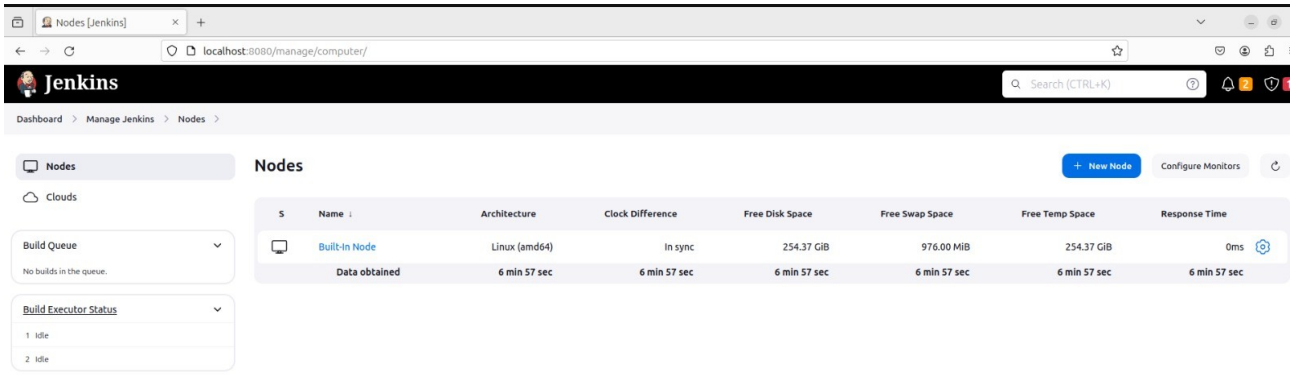
Console output



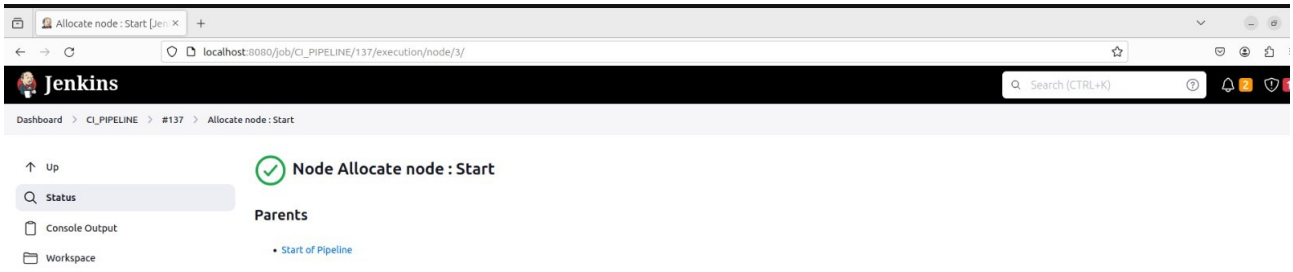
Pipeline Steps



Node Details.



Node Build Details.



References:

https://www.jenkins.io/blog/2022/06/28/require-java-11/

https://www.jenkins.io/doc/administration/requirements/upgrade-java-guidelines/#:~:text=If%20you're%20upgrading%20your,screen%20of%20your%20Jenkins%20instance.

https://stackoverflow.com/questions/69495517/unable-to-install-jenkins-on-ubuntu-20-04

https://stackoverflow.com/questions/14119983/java-home-and-path-are-set-but-java-versionstill-shows-the-old-one

Task 3:

Write a Docket file. Create an Image and container on the Docker host. Integrate docker host with Jenkins. Create CI/CD job on Jenkins to build and deploy on a container.

1. **Enhance the package job created in step 1 of task 2 to create a docker image.**
2. **In the Docker image, add code to move the war file to the Tomcat server and build the image**

Approach i have followed:

Step 1: Created a Docker file

Created a Docker file in the root directory. The file has the instructions for building Docker image.

Docker file

Code Blame 4 lines (3 loc) · 130 Bytes Code 55% faster with GitHub Copilot Raw

```
FROM tomcat:latest
RUN cp -R  /usr/local/tomcat/webapps.dist/*  /usr/local/tomcat/webapps
COPY ./*.war /usr/local/tomcat/webapps
```

This Docker file installed Docker inside the Jenkins image and copied Jenkins job configurations from the jobs directory to the appropriate location inside the container.

Step 2: Built the Docker Image

Built the Docker image using the following command within the Jenkinsfile:

```
        stage('Build Docker Image') {
          steps {
            sh 'sudo docker build -t ${DOCKER_HUB_REPO}:$BUILD_NUMBER .'
            echo 'Build Image Completed'
          }
        }

```

Step 3: Ran the Docker Container

Ran the Docker container :

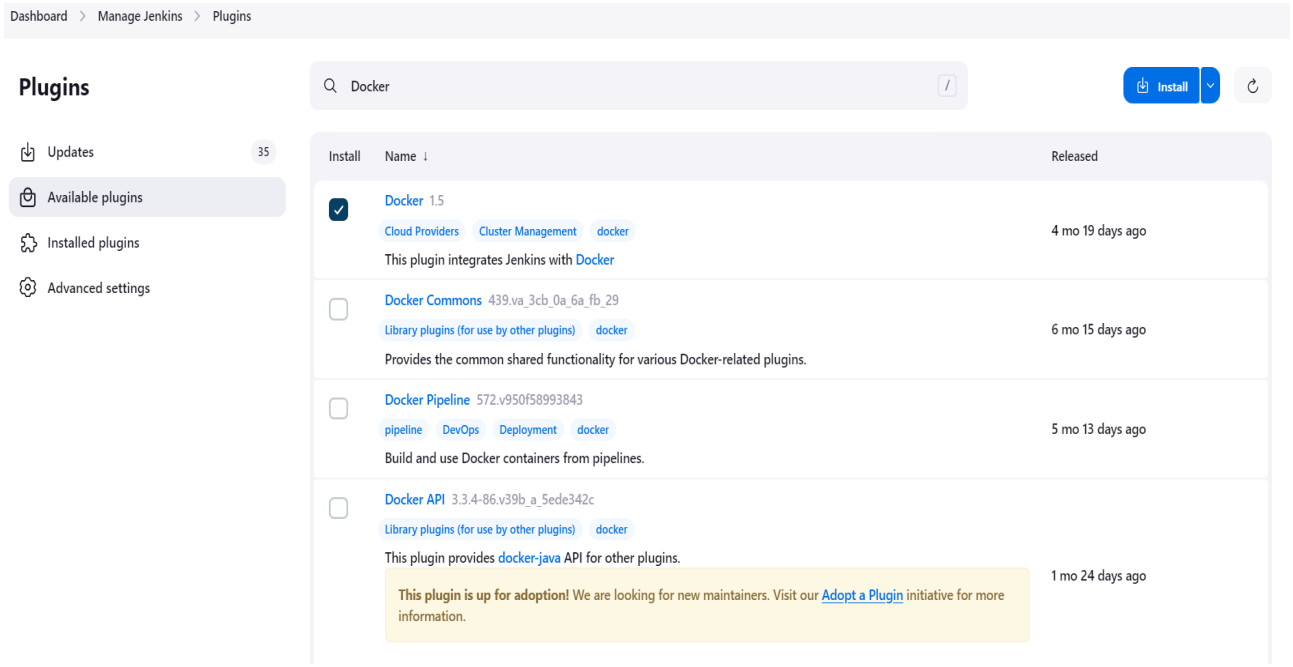
```
---
- hosts: ansible-server
  become: true

  tasks:

  - name: Login to Dockerhub
    command: sudo docker login -u bejoykoottumkal@gmail.com -p dckr_pat_KKj6tEX1WMiDnqPcA3olJMd7kos

  - name: create docker image using war file
    command: sudo docker build -t bejoykoottumkal/devops-lab:latest .
    args:
      chdir: /opt/k8s-lab

  - name: create tag to image
    command: sudo docker tag bejoykoottumkal/devops-lab bejoykoottumkal/devops-lab

  - name: push image on to dockerhub
    command: sudo docker push bejoykoottumkal/devops-lab:latest
```

Mapped the Docker socket into the container, allowing Jenkins to communicate with the Docker daemon on the host.

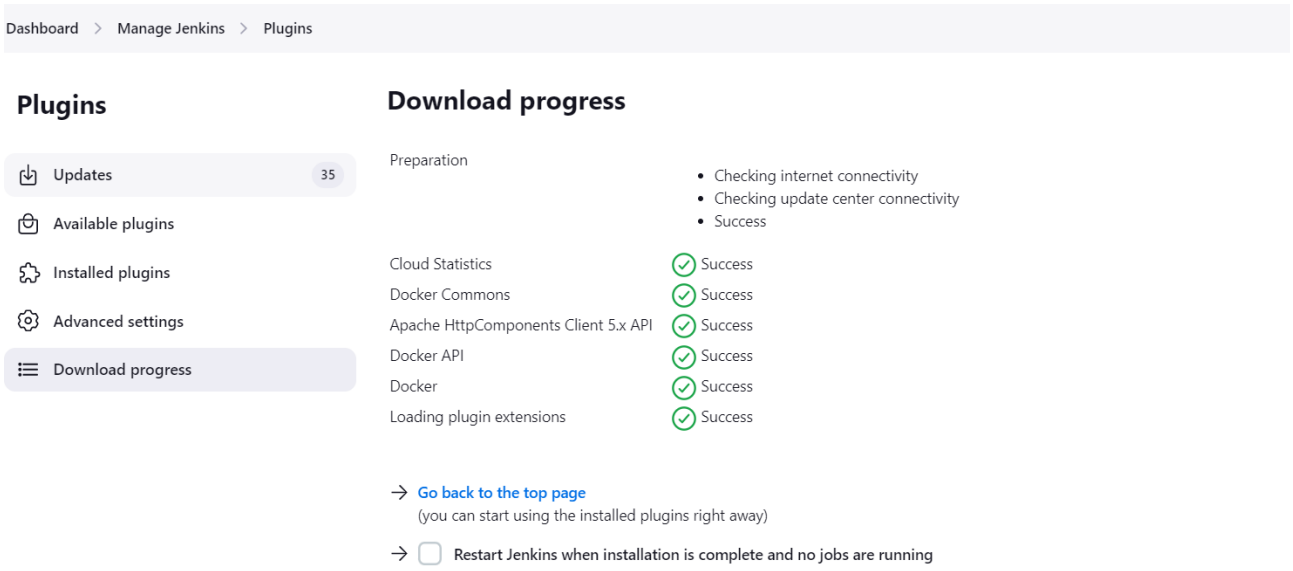
Step 4: Integrated Docker Host with Jenkins

In the Jenkins dashboard, navigated to "Manage Jenkins" -> "Manage Plugins" -> "Available."

Searched for and installed the "Docker" plugin.

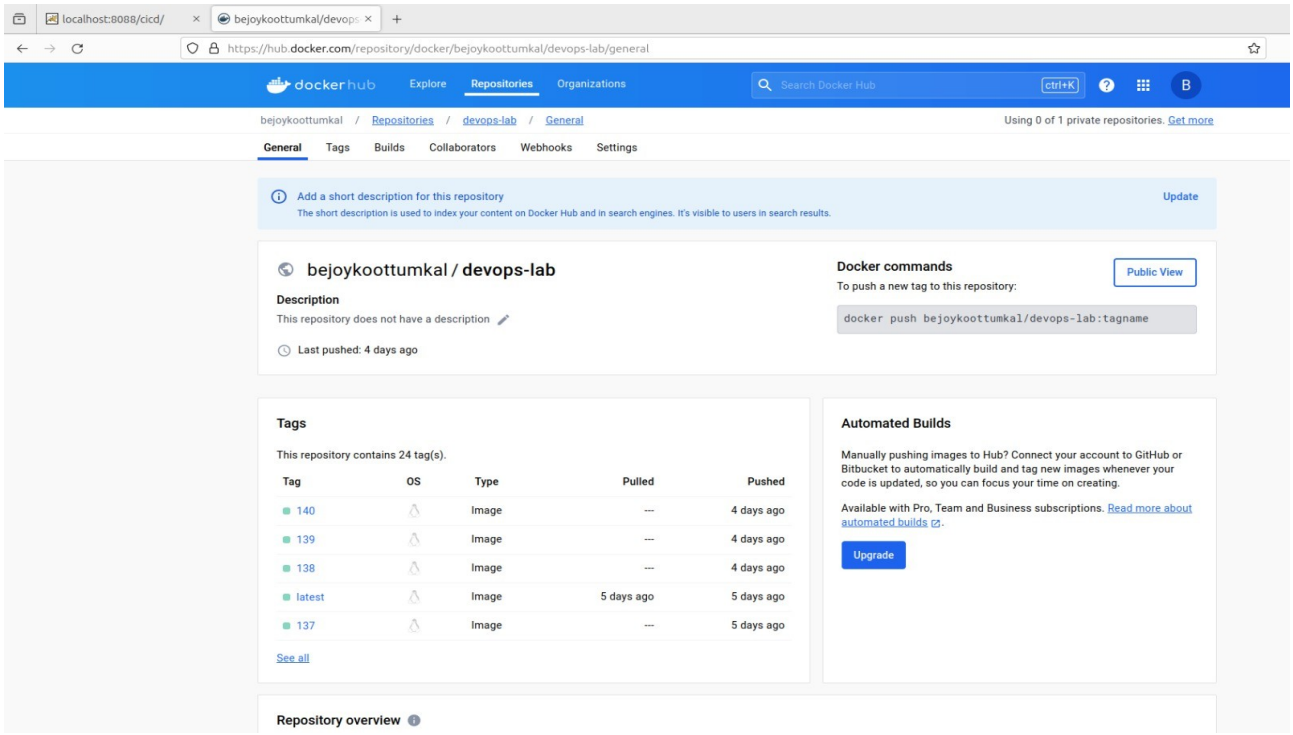


Docker Installation Complete



Step 5: Created an account in the Docker Hub

Created a repository : bejoykoottumkal/devops-lab



Step 6: Login to Docker HUB

Using the pipeline script created a stage for login to docker hub account

```
stage('Login to Docker Hub') {
  steps{
    sh 'echo $DOCKERHUB_CREDENTIALS_PSW | sudo docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
    echo 'Login Completed'
  }
}
```

Step 7: Pushed the image to the Docker Hub

```
stage('Push Image to Docker Hub') {
  steps {
    sh 'sudo docker push ${DOCKER_HUB_REPO}:$BUILD_NUMBER'
    echo 'Push Image Completed'
  }
}
```

References:

https://plugins.jenkins.io/role-strategy/

https://stackoverflow.com/questions/37603621/jenkins-sudo-no-tty-present-and-no-askpass#program-specified-with-nopasswd

https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.68/bin/apache-tomcat-9.0.68.tar.gz

https://phoenixnap.com/kb/how-to-configure-docker-in-jenkins

https://www.tutorialspoint.com/docker/docker_continuous_integration.htm

https://www.provartesting.com/documentation/devops/continuous-integration/docker/setting#up-continuous-integration-with-jenkins-for-docker/

https://www.youtube.com/watch?v=mszE-OCI2V4&list=PLVz2XdJiJQxwS0BZUHX34ocLTJtRGSQzN&index=5

https://www.digitalocean.com/community/questions/how-to-fix-docker-got-permission-denied-while-trying-to-connect-to-the-docker-daemon-socket

https://stackoverflow.com/questions/50798720/jenkins-throwing-error-jenkins-model-invalidbuildsdir-item-rootdir-builds-d

https://linuxize.com/post/how-to-list-groups-in-linux/

https://stackoverflow.com/questions/17733671/how-can-i-tell-what-user-jenkins-is-running-as

https://stackoverflow.com/questions/55156958/jenkins-fail-to-deploy-war-to-tomcat-container-second-time

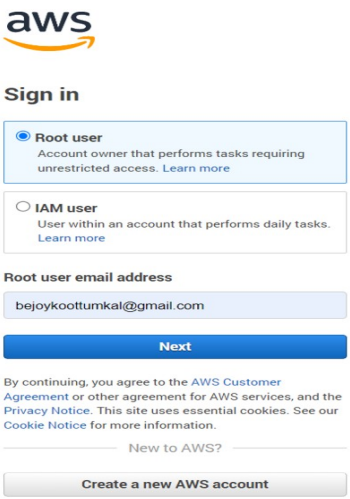
Task 4:

Integrate Docker host with Ansible. Write ansible playbook to create Image and create continuer. Integrate Ansible with Jenkins. Deploy ansible-playbook. CI/CD job to build code on ansible and deploy it on docker container

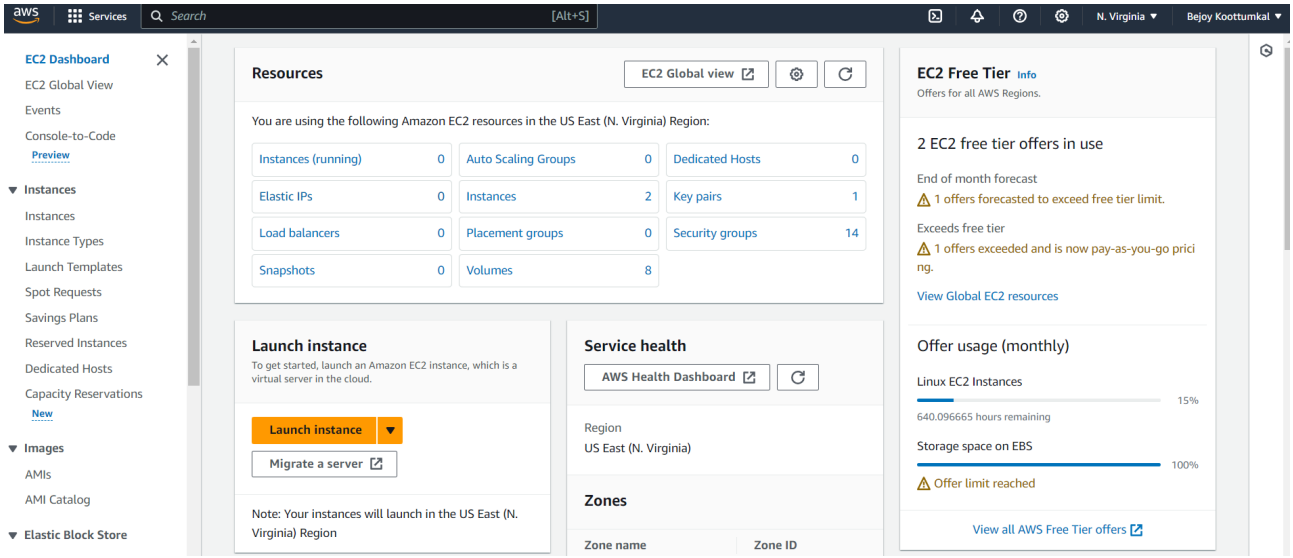
1. **Deploy Artifacts on Kubernetes**
2. **Write pod, service, and deployment manifest file**
3. **Integrate Kubernetes with ansible**
4. **Ansible playbook to create deployment and service**

Approach i have followed:

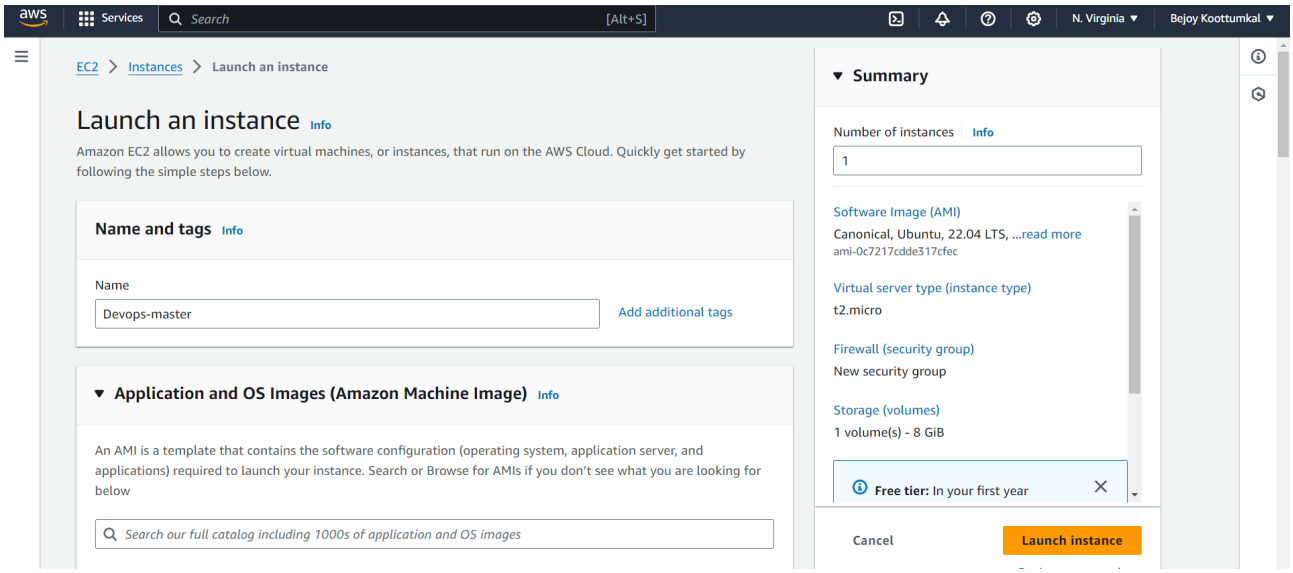
Step1: Login to AWS



EC2 Dashboard.



Step 2: Created an EC2 Instance with t2 micro- devops-master.



This instance was used for launching Kubernetes Master and Client nodes

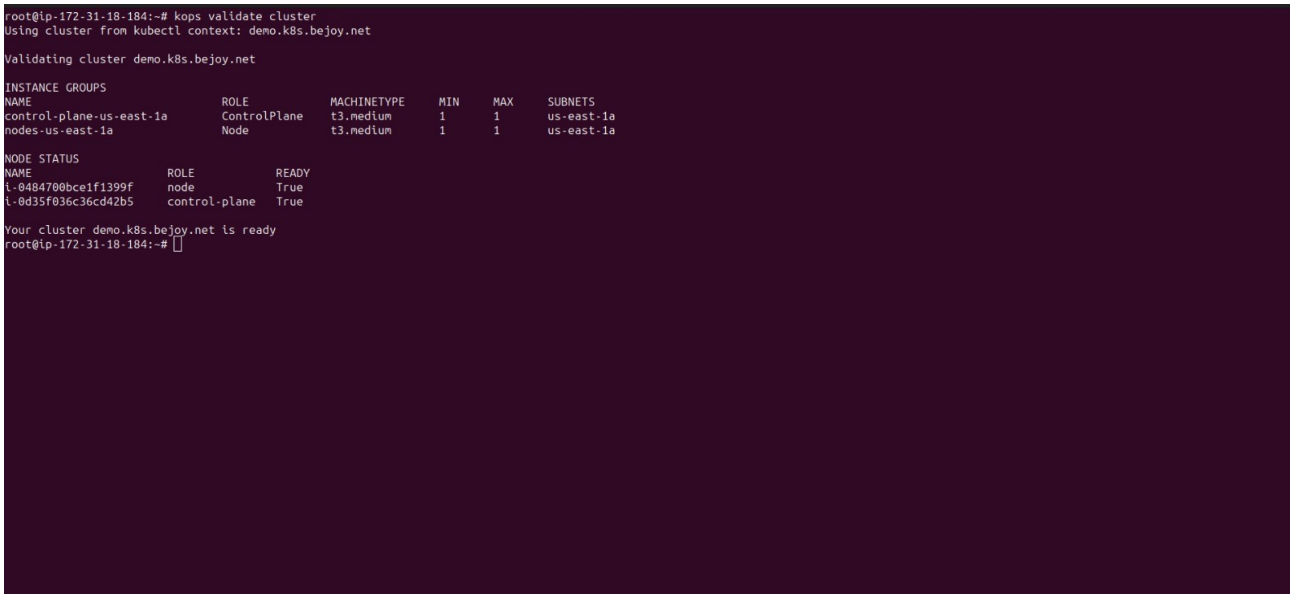
Step 3: Set Up Master and Client Nodes:

I followed Kubernetes Installation using the following Link

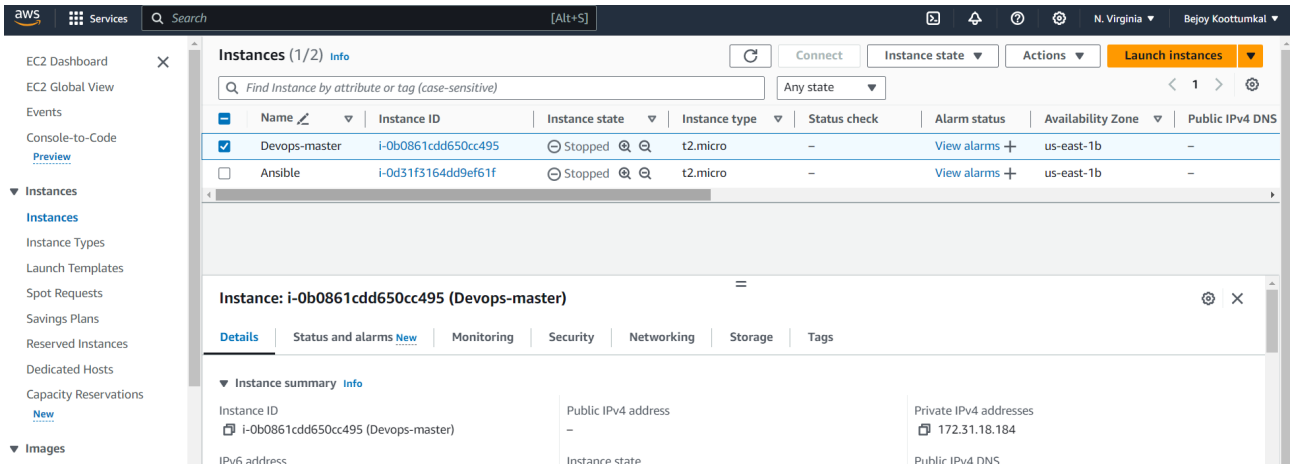
Reference:

https://github.com/yankils/Simple-DevOps-Project/blob/master/Kubernetes Kubernetes_Setup_using_kops.md

After the installation, i verified the kubernetes master and client node status using the command: # kops validate cluster



Step 4: From AWS EC2 Console launched another machine for Ansible Configuration.



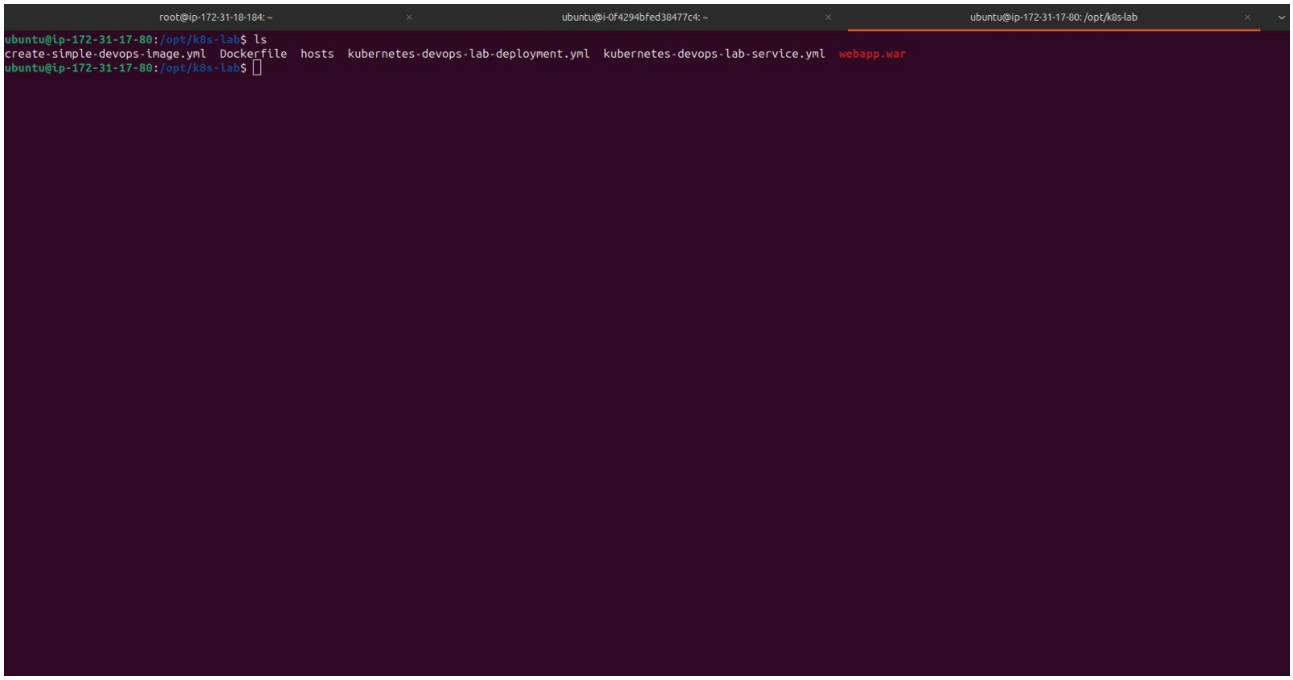
Reference:

https://www.cherryservers.com/blog/install-ansible-ubuntu

After the installation created a work folder for Ansible**: /opt/k8s-lab**

Copied the files from **:** https://github.com/bejoykoottumkal/devops-lab/tree/main/cicd/ansible

to the ansible work folder.



Step 5: Copied the deploy and service files to K8s master.

SSH to K8s Master and copied the files from:

https://github.com/bejoykoottumkal/devops-lab/tree/main/cicd/k8s

To

/home/ubuntu

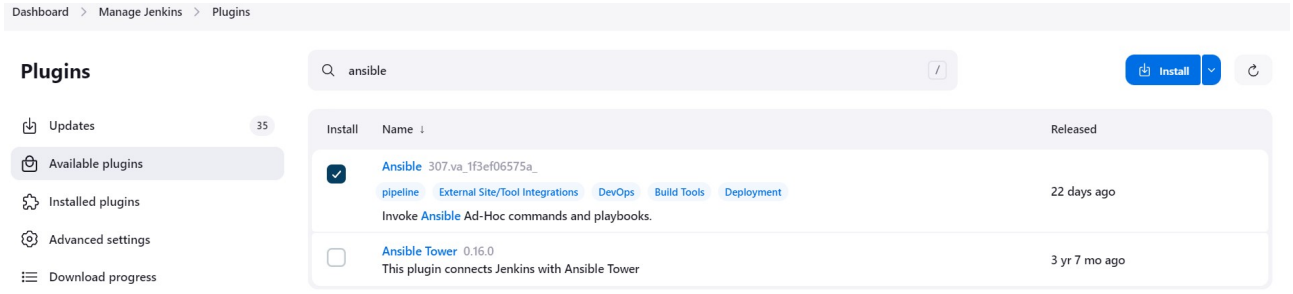
Step 6: Integrated Ansible with Jenkins

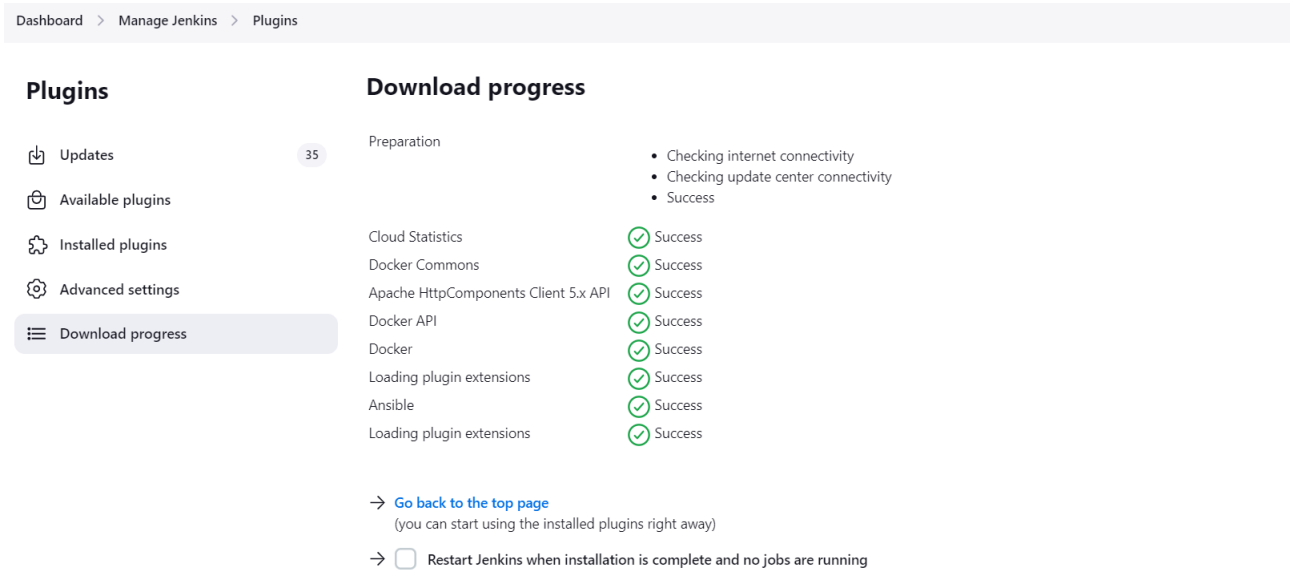
The Ansible plugin was installed on Jenkins through the following steps:

Navigated to Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available.

Searched for "Ansible" and installed the plugin.

Ansible was configured in Jenkins:





Step 7: Deployed ansible-playbook through pipeline script.

```
stage('Deploy to K8s') {
    steps {
        sshagent(credentials: ['ssh_ansible']) {
          sh 'scp $WORKSPACE/webapp/target/webapp.war ubuntu@$ANSIBLE_SERVER_IP:/opt/k8s-lab/'
        }
        sshagent(credentials: ['ssh_ansible']) {
          sh 'ssh -o StrictHostKeyChecking=no -l ubuntu $ANSIBLE_SERVER_IP ansible-playbook -i /opt/k8s-lab/hosts /opt/k8s-lab/create-simple-devops-image.yml'
        }
        sshagent(credentials: ['ssh_ansible']) {
          sh 'ssh -o StrictHostKeyChecking=no -l ubuntu $ANSIBLE_SERVER_IP ansible-playbook -i /opt/k8s-lab/hosts /opt/k8s-lab/kubernetes-devops-lab-deployment.yml'
        }
        sshagent(credentials: ['ssh_ansible']) {
          sh 'ssh -o StrictHostKeyChecking=no -l ubuntu $ANSIBLE_SERVER_IP ansible-playbook -i ansible-playbook -i /opt/k8s-lab/hosts /opt/k8s-lab/kubernetes-devops-lab-service.yml'
        }
    }
}
```

Step 8: Deployed Artifacts on Kubernetes

Kubernetes manifest files were written to define Deployment and Service configurations.

An Ansible playbook for Kubernetes deployment, cicd/k8s/devops-lab-deploy.yml, was created:

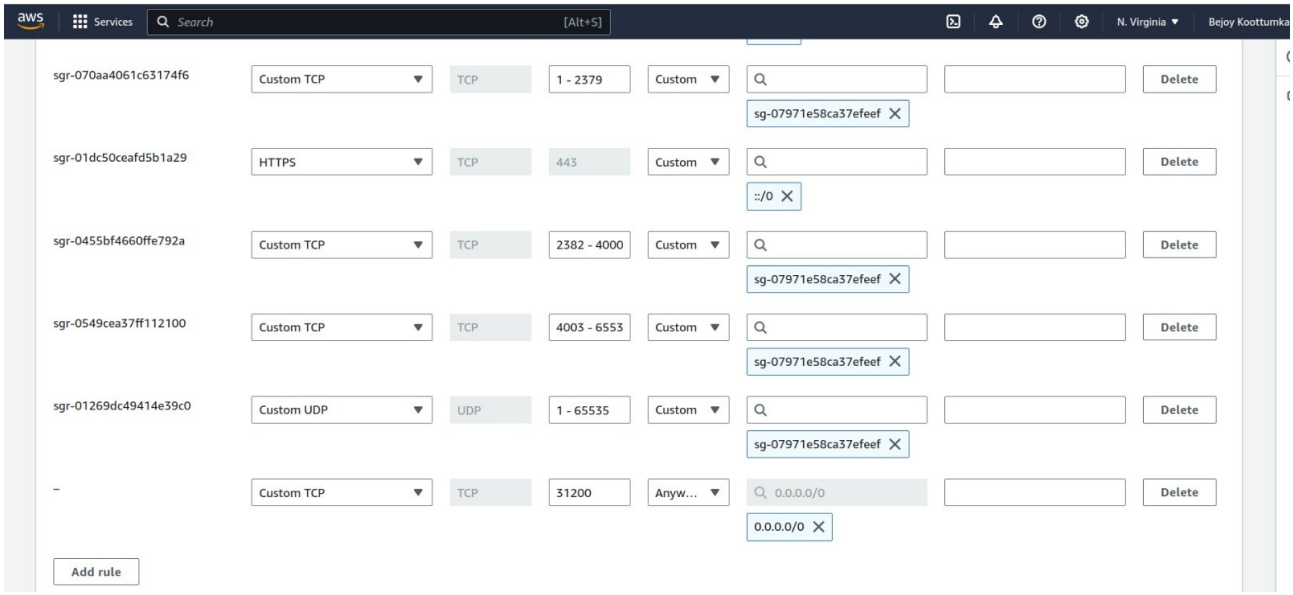
Code Blame 27 lines (25 loc) · 628 Bytes Code 55% faster with GitHub Copilot Raw

```yaml
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: devops-lab-deployment
spec:
  selector:
    matchLabels:
      app: devops-lab-devops-project
  replicas: 2 # tells deployment to run 2 pods matching the template
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1

  template:
    metadata:
      labels:
        app: devops-lab-devops-project
    spec:
      containers:
      - name: devops-lab-devops-project
        image: bejoykoottumkal/devops-lab
        imagePullPolicy: Always
        ports:
        - containerPort: 8080
```

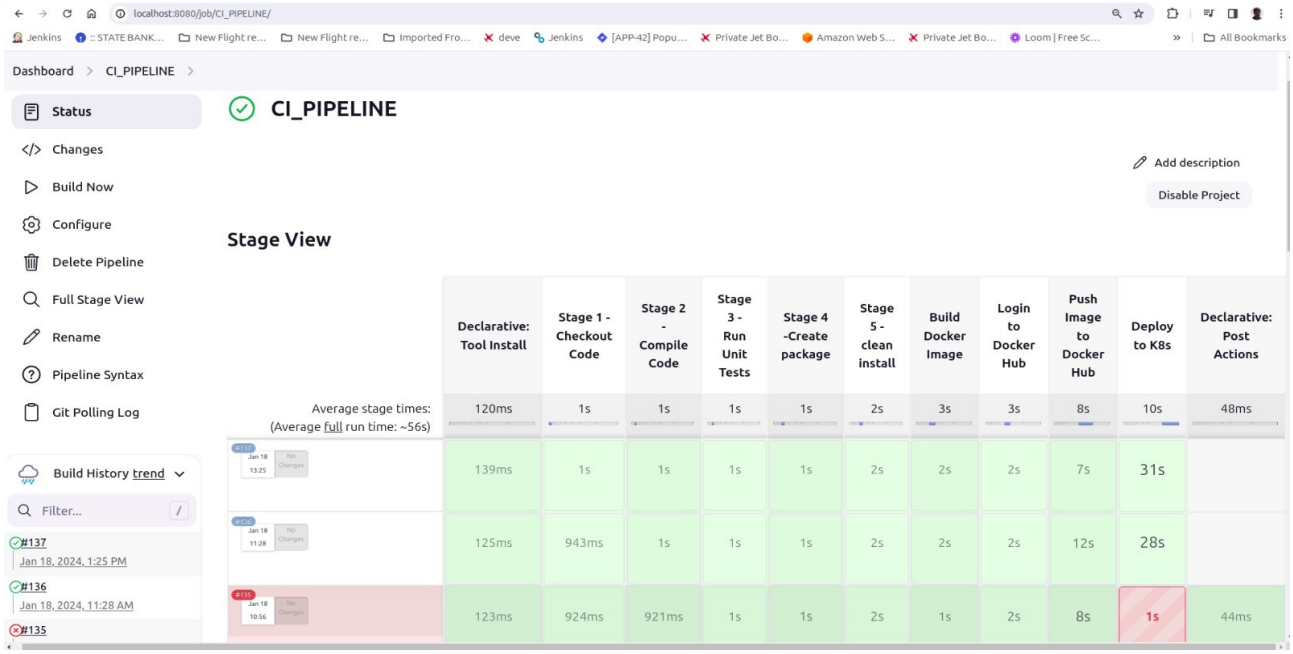
```yaml
apiVersion: v1
kind: Service
metadata:
  name: devops-lab-service
  labels:
    app: devops-lab-devops-project
spec:
  selector:
    app: devops-lab-devops-project
  type: LoadBalancer
  ports:
    - port: 8080
      targetPort: 8080
      nodePort: 31200
```

Step 8: White listing Application Port in AWS Security group

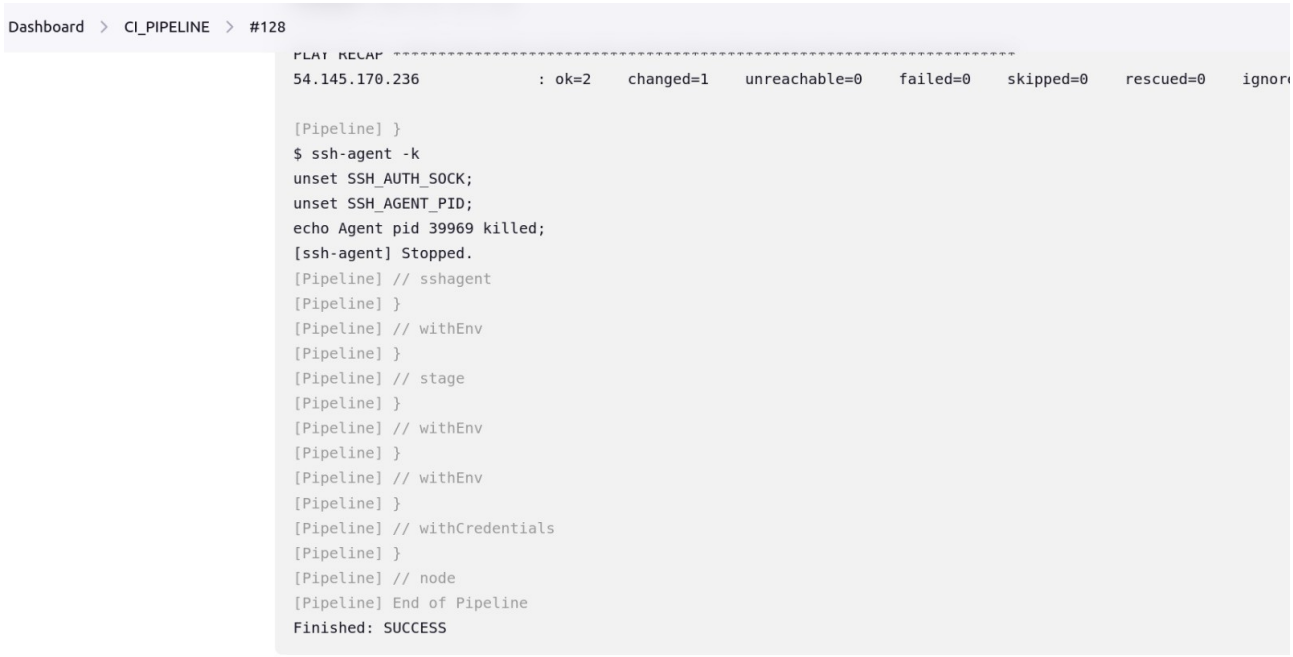
Enabling 31200 in AWS security group



Step 9: Running the CICD Pipeline by doing a Test Commit and Pushed to master branch.

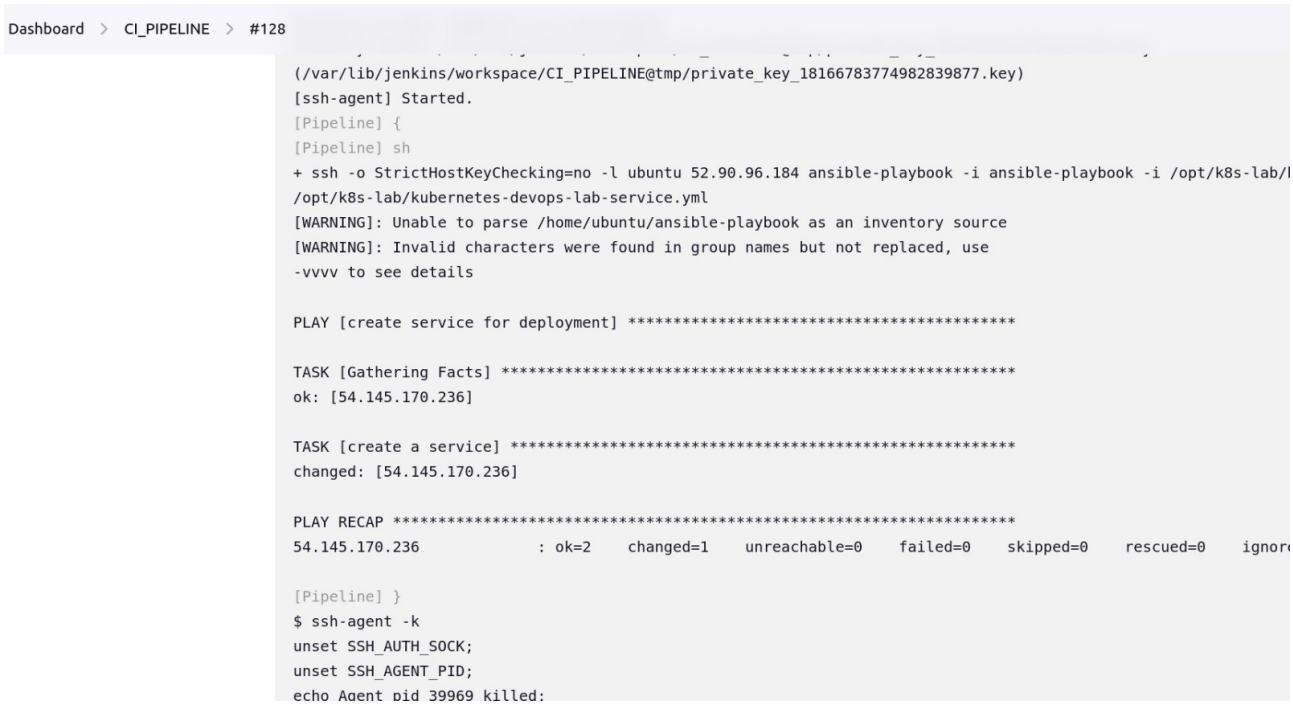


Console Output:



Dashboard > CI_PIPELINE > #128

```
PLAY RECAP *********************************************************************
54.145.170.236             : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignore

[Pipeline] }
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 39969 killed;
[ssh-agent] Stopped.
[Pipeline] // sshagent
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



Dashboard > CI_PIPELINE > #128

```
(/var/lib/jenkins/workspace/CI_PIPELINE@tmp/private_key_18166783774982839877.key)
[ssh-agent] Started.
[Pipeline] {
[Pipeline] sh
+ ssh -o StrictHostKeyChecking=no -l ubuntu 52.90.96.184 ansible-playbook -i ansible-playbook -i /opt/k8s-lab/
/opt/k8s-lab/kubernetes-devops-lab-service.yml
[WARNING]: Unable to parse /home/ubuntu/ansible-playbook as an inventory source
[WARNING]: Invalid characters were found in group names but not replaced, use
-vvvv to see details

PLAY [create service for deployment] *******************************************

TASK [Gathering Facts] *********************************************************
ok: [54.145.170.236]

TASK [create a service] ********************************************************
changed: [54.145.170.236]

PLAY RECAP *********************************************************************
54.145.170.236             : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignore

[Pipeline] }
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 39969 killed;
```

Dashboard > CI_PIPELINE > #128

```
[Pipeline] sh
+ sudo docker build -t bejoykoottumkal/devops-lab:128 .
#0 building with "default" instance using docker driver

#1 [internal] load .dockerignore
#1 transferring context: 2B done
#1 DONE 0.0s

#2 [internal] load build definition from Dockerfile
#2 transferring dockerfile: 169B done
#2 DONE 0.0s

#3 [internal] load metadata for docker.io/library/tomcat:latest
#3 ...

#4 [auth] library/tomcat:pull token for registry-1.docker.io
#4 DONE 0.0s

#3 [internal] load metadata for docker.io/library/tomcat:latest
#3 DONE 2.8s

#5 [1/3] FROM docker.io/library/tomcat:latest@sha256:504823cf27f257059521ba0b9d40e88761109fd78f6b7bc3212747346a
#5 DONE 0.0s

#6 [internal] load build context
#6 transferring context: 2B 0.0s done
```

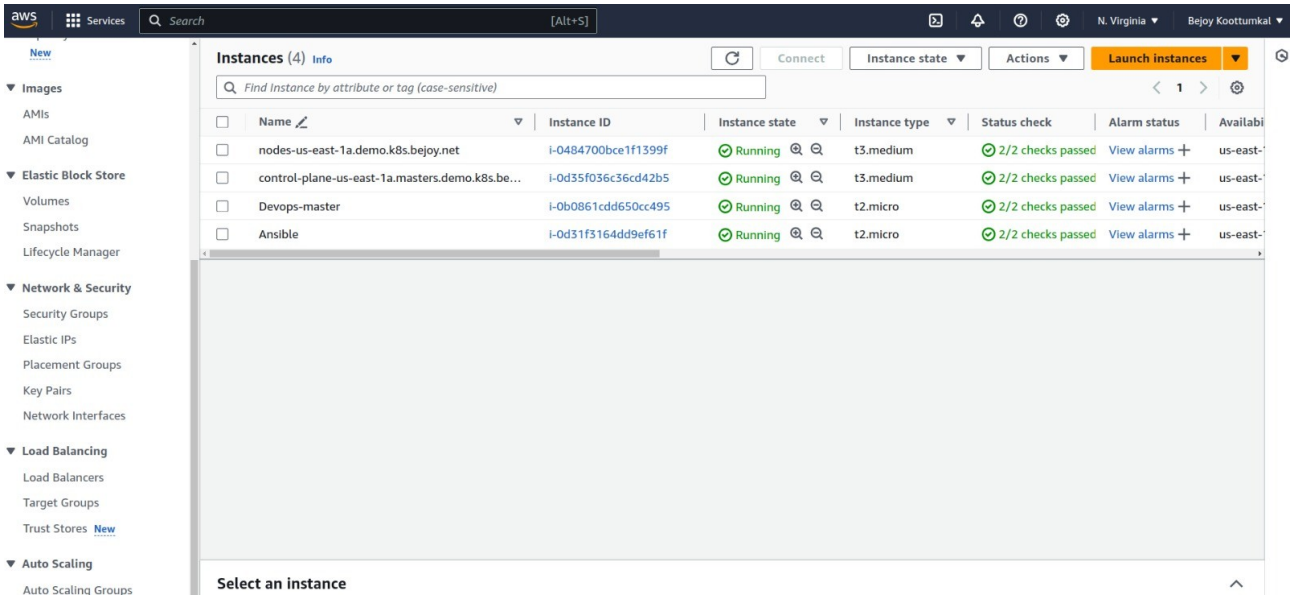
Dashboard > CI_PIPELINE > #128

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------

Results :

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for Maven Project 1.0-SNAPSHOT:
[INFO]
[INFO] Maven Project ..................................... SUCCESS [  0.001 s]
[INFO] Server ............................................ SUCCESS [  0.486 s]
[INFO] Webapp ............................................ SUCCESS [  0.088 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.629 s
[INFO] Finished at: 2024-01-16T18:29:13+05:30
[INFO] ------------------------------------------------------------------------
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
```

Dashboard > CI_PIPELINE > #128

```
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] ------------------< com.example.maven-project:webapp >------------------
[INFO] Building Webapp 1.0-SNAPSHOT                                       [3/3]
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.5:resources (default-resources) @ webapp ---
[debug] execute contextualize
[INFO] Using 'utf-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/CI_PIPELINE/webapp/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ webapp ---
[INFO] No sources to compile
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for Maven Project 1.0-SNAPSHOT:
[INFO]
[INFO] Maven Project ..................................... SUCCESS [  0.001 s]
[INFO] Server ............................................ SUCCESS [  0.246 s]
[INFO] Webapp ............................................ SUCCESS [  0.010 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.311 s
[INFO] Finished at: 2024-01-16T18:29:12+05:30
[INFO] ------------------------------------------------------------------------
[Pipeline] }
```

AWS Instances Up and Running.



Application Live Screenshot:

Application up and running in http://54.166.216.219:31200/webapp/

References:

https://docs.ansible.com/ansible/latest/collections/community/docker/docker_image_module.Html

https://docs.ansible.com/ansible/latest/collections/community/dockerdocker_container_module.html

https://github.com/ansible-collections/community.docker

https://pypi.org/project/docker/

https://pypi.org/project/docker-py/

https://github.com/ansible-collections/community.docker

https://docker-py.readthedocs.io/en/latest/

https://www.baeldung.com/ops/docker-removing-images#:~:text=Forcefully%20Remove%20Containers%20and%20Images&text=The%20%2Df%20flag%20is%20used,the%20running%20Docker%20containers%20forcefully.&text=The%20docker%20images%20%2Dqa%20will,forcefully%20remove%20the%20Docker%20image

https://stackoverflow.com/questions/22769568/system-specific-variables-in-ansible

https://stackoverflow.com/questions/60834692/how-to-completely-remove-ansible-2-8-3-on-ubuntu-18-04--

https://docs.ansible.com/ansible/2.6/modules/k8s_module.html#examples

https://docs.ansible.com/ansible/latest/user_guide/guide_rolling_upgrade.html

https://adamtheautomator.com/ansible-kubernetes/

https://adamtheautomator.com/ansible-template/ansible-playbook%20sample-playbook.yml%20-e%20'ansible_python_interpreter=/usr/bin/python3'%20-

https://docs.ansible.com/ansible/latest/reference_appendices/python_3_support.html

https://docs.ansible.com/ansible/2.9/modules/k8s_service_module.html

https://www.youtube.com/watch?v=NSk0NHkTjDs

https://www.baeldung.com/ops/jenkins-parameterized-builds

https://stackoverflow.com/questions/56003777/how-to-pass-environment-variable-in-kubectl#deployment

https://stackoverflow.com/questions/62108860/kubectl-no-matches-for-kind-service-in-version-apps-v1

https://stackoverflow.com/questions/59579482/kubectl-apply-error-from-server-forbidden-authentication-required-jenkins

https://stackoverflow.com/questions/51489359/docker-using-password-via-the-cli-is-insecure-use-password-stdin

https://kubernetes.io/docs/reference/kubectl/quick-reference/

Task 5:

Using Prometheus monitor the resources like CPU utilization: Total Usage, Usage per core, usage breakdown, Memory , Network on the instance by providing the end points in local host. Install node exporter and add URL to target in Prometheus. Using this data login to Grafana and create a dashboard to show the metrics

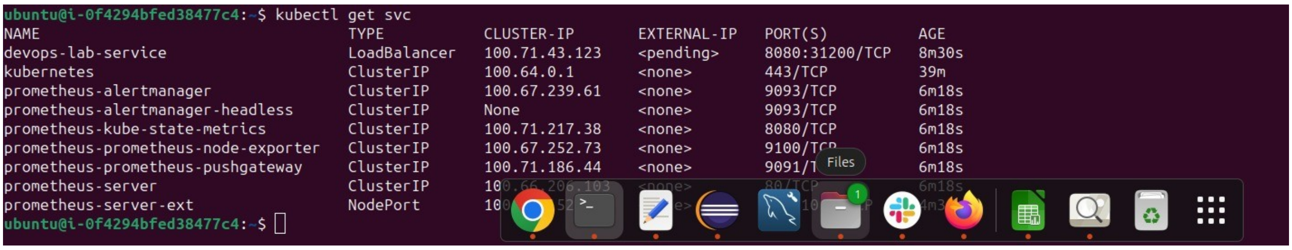
Approach i have followed:

Step 1: Installed Prometheus

- **SSH to Kubernates Master**
- **Installed Prometheus**
- **Find the port listening for prometheus**
- **Prometheus-server-ext => 31091**

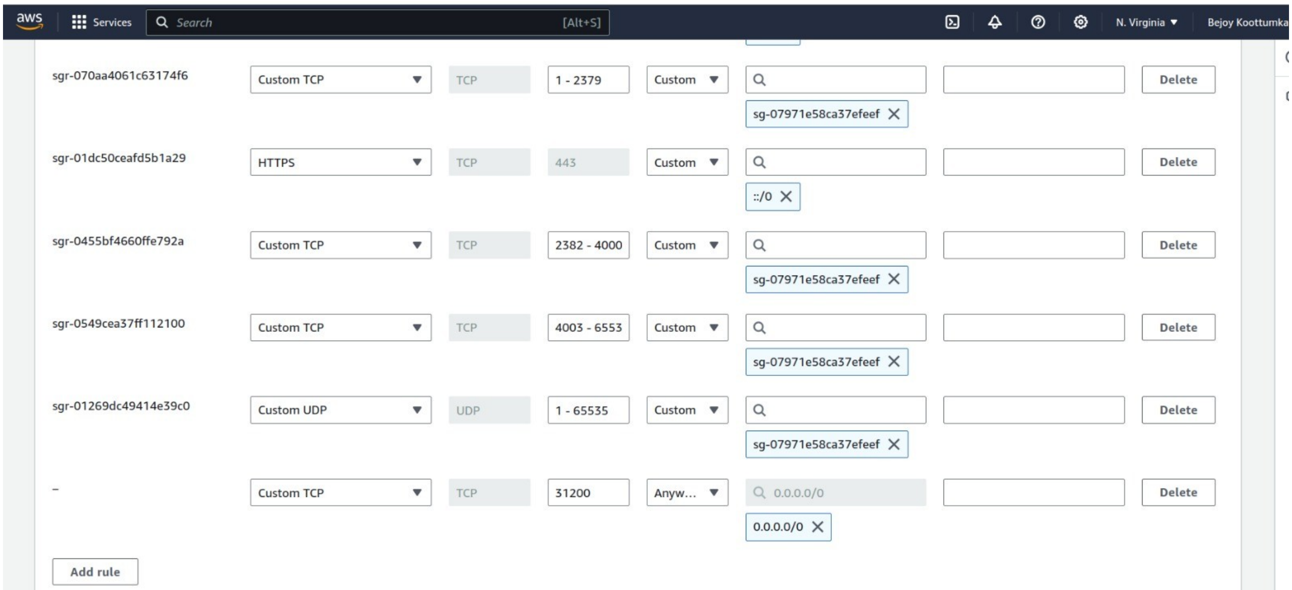
```
https://prometheus.io/
ubuntu@i-0f4294bfed38477c4:~$ kubectl get pods
NAME                                             READY   STATUS    RESTARTS   AGE
devops-lab-deployment-6cd44464c8-498qm           1/1     Running   0          2m44s
devops-lab-deployment-6cd44464c8-89jxc           1/1     Running   0          2m44s
prometheus-alertmanager-0                        1/1     Running   0          25s
prometheus-kube-state-metrics-6b464f5b88-wpqws   1/1     Running   0          25s
prometheus-prometheus-node-exporter-q56xq        1/1     Running   0          25s
prometheus-prometheus-node-exporter-x69ns        1/1     Running   0          25s
prometheus-prometheus-pushgateway-7857c44f49-zwj9x   1/1     Running   0          25s
prometheus-server-6b68fbd54b-swrjn               1/2     Running   0          25s
ubuntu@i-0f4294bfed38477c4:~$ kubectl get svc
NAME                                TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
devops-lab-service                  LoadBalancer   100.71.43.123    <pending>     8080:31200/TCP   2m49s
kubernetes                          ClusterIP      100.64.0.1       <none>        443/TCP          34m
prometheus-alertmanager             ClusterIP      100.67.239.61    <none>        9093/TCP         37s
prometheus-alertmanager-headless    ClusterIP      None             <none>        9093/TCP         37s
prometheus-kube-state-metrics       ClusterIP      100.71.217.38    <none>        8080/TCP         37s
prometheus-prometheus-node-exporter   ClusterIP      100.67.252.73    <none>        9100/TCP         37s
prometheus-prometheus-pushgateway   ClusterIP      100.71.186.44    <none>        9091/TCP         37s
prometheus-server                   ClusterIP      100.66.206.103   <none>        80/TCP           37s
ubuntu@i-0f4294bfed38477c4:~$ kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext
service/prometheus-server-ext exposed
ubuntu@i-0f4294bfed38477c4:~$
```

```
root@ip-172-31-18-184:~# kubectl get pods
NAME                                             READY   STATUS              RESTARTS   AGE
devops-lab-deployment-8674699c99-g9bzd           1/1     Running             0          6m11s
devops-lab-deployment-8674699c99-x2xbv           1/1     Running             0          6m11s
prometheus-alertmanager-0                        0/1     ContainerCreating   0          99m
prometheus-kube-state-metrics-6b464f5b88-g7hm4   1/1     Running             0          99m
prometheus-prometheus-node-exporter-f2kft        1/1     Running             0          98m
prometheus-prometheus-node-exporter-wdz49        1/1     Running             0          101m
prometheus-prometheus-pushgateway-7857c44f49-2gz8w   1/1     Running             0          99m
prometheus-server-6b68fbd54b-bdz72               0/2     ContainerCreating   0          99m
root@ip-172-31-18-184:~# kubectl logs --tail=50 devops-lab-deployment-8674699c99-g9bzd
18-Jan-2024 05:59:24.748 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version name:   Apache Tomcat/10.1.18
18-Jan-2024 05:59:24.758 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built:          Jan 5 2024 14:39:40 UTC
18-Jan-2024 05:59:24.759 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version number: 10.1.18.0
18-Jan-2024 05:59:24.759 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name:               Linux
18-Jan-2024 05:59:24.759 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Version:            6.2.0-1017-aws
18-Jan-2024 05:59:24.760 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Architecture:          amd64
18-Jan-2024 05:59:24.760 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home:             /opt/java/openjdk
18-Jan-2024 05:59:24.762 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Version:          21.0.1+12-LTS
18-Jan-2024 05:59:24.762 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Vendor:           Eclipse Adoptium
18-Jan-2024 05:59:24.762 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_BASE:        /usr/local/tomcat
18-Jan-2024 05:59:24.763 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_HOME:        /usr/local/tomcat
18-Jan-2024 05:59:24.797 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=/usr/local/tomcat/conf/logging.properties
18-Jan-2024 05:59:24.808 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
18-Jan-2024 05:59:24.820 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
18-Jan-2024 05:59:24.821 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=org.apache.catalina.webresources
18-Jan-2024 05:59:24.821 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dorg.apache.catalina.security.SecurityListener.UMASK=0027
18-Jan-2024 05:59:24.821 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.lang=ALL-UNNAMED
18-Jan-2024 05:59:24.821 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.io=ALL-UNNAMED
18-Jan-2024 05:59:24.821 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.util=ALL-UNNAMED
18-Jan-2024 05:59:24.821 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.base/java.util.concurrent=ALL-UNNAMED
18-Jan-2024 05:59:24.821 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
18-Jan-2024 05:59:24.822 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dcatalina.base=/usr/local/tomcat
18-Jan-2024 05:59:24.822 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dcatalina.home=/usr/local/tomcat
18-Jan-2024 05:59:24.822 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.io.tmpdir=/usr/local/tomcat/temp
18-Jan-2024 05:59:24.845 INFO [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent Loaded Apache Tomcat Native library [2.0.6] using APR version [1.7.0].
18-Jan-2024 05:59:24.853 INFO [main] org.apache.catalina.core.AprLifecycleListener.initializeSSL OpenSSL successfully initialized [OpenSSL 3.0.2 15 Mar 2022]
18-Jan-2024 05:59:26.015 INFO [main] org.apache.coyote.AbstractProtocol.init Initializing ProtocolHandler ["http-nio-8080"]
18-Jan-2024 05:59:26.082 INFO [main] org.apache.catalina.startup.Catalina.load Server initialization in [2224] milliseconds
18-Jan-2024 05:59:26.217 INFO [main] org.apache.catalina.core.StandardService.startInternal Starting service [Catalina]
18-Jan-2024 05:59:26.218 INFO [main] org.apache.catalina.core.StandardEngine.startInternal Starting Servlet engine: [Apache Tomcat/10.1.18]
18-Jan-2024 05:59:26.306 INFO [main] org.apache.catalina.startup.HostConfig.deployWAR Deploying web application archive [/usr/local/tomcat/webapps/webapp.war]
18-Jan-2024 05:59:27.173 INFO [main] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive [/usr/local/tomcat/webapps/webapp.war] has finished in [866] ms
18-Jan-2024 05:59:27.183 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
18-Jan-2024 05:59:27.201 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [1112] milliseconds
root@ip-172-31-18-184:~#
```



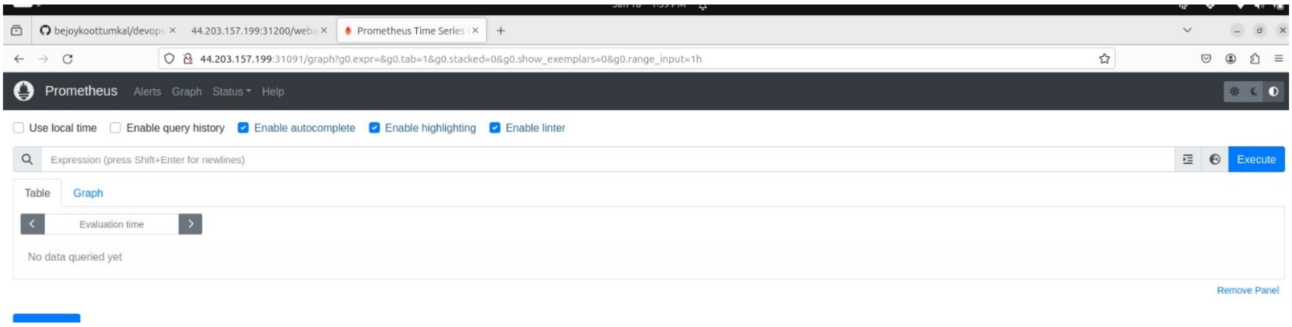
Step 2 :White listing Prometheus Port in AWS Security group

- **Open port 31091 in AWS security group**

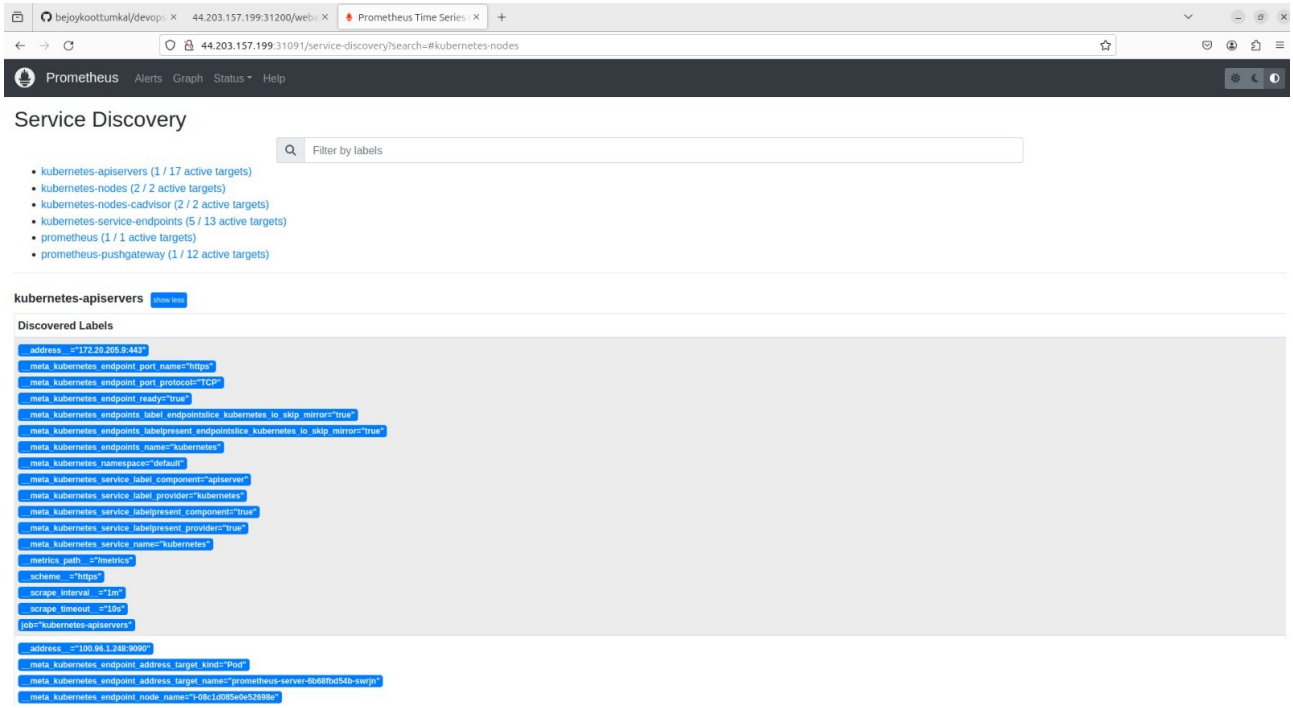


Step 3: Prometheus Monitoring

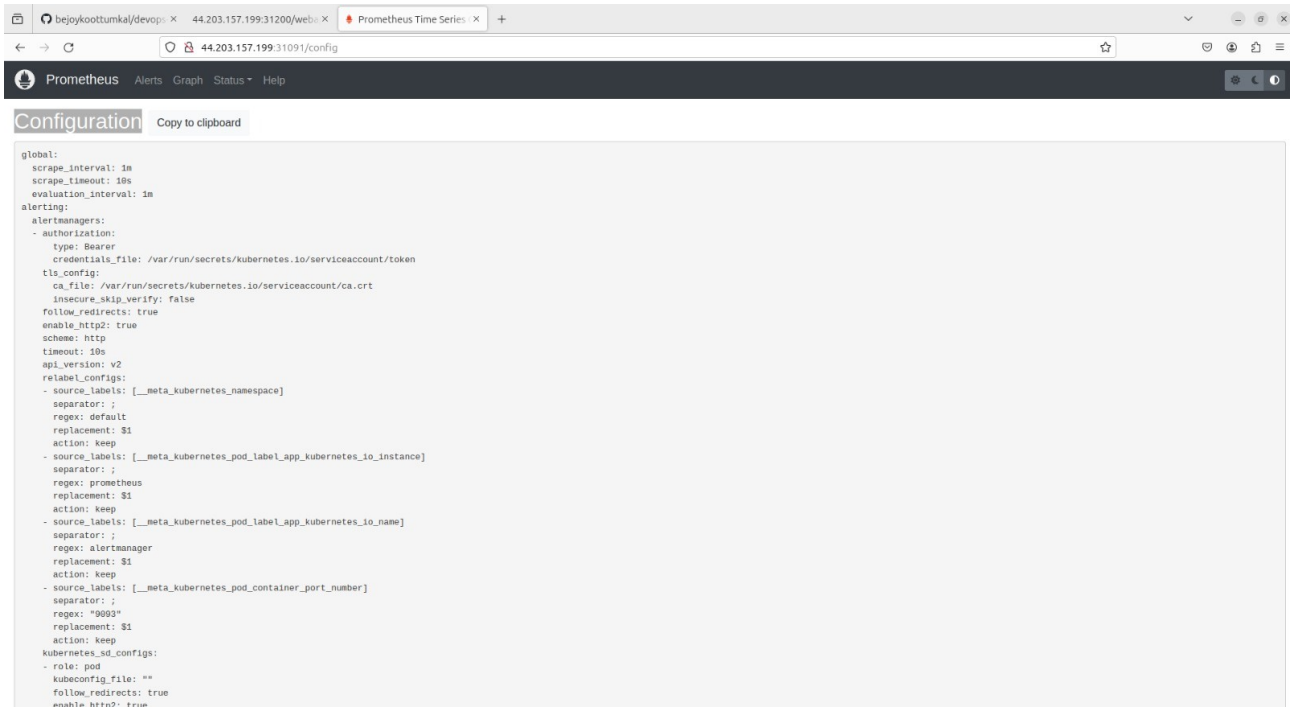
Prometheus started at url - http://44.203.157.199:31091



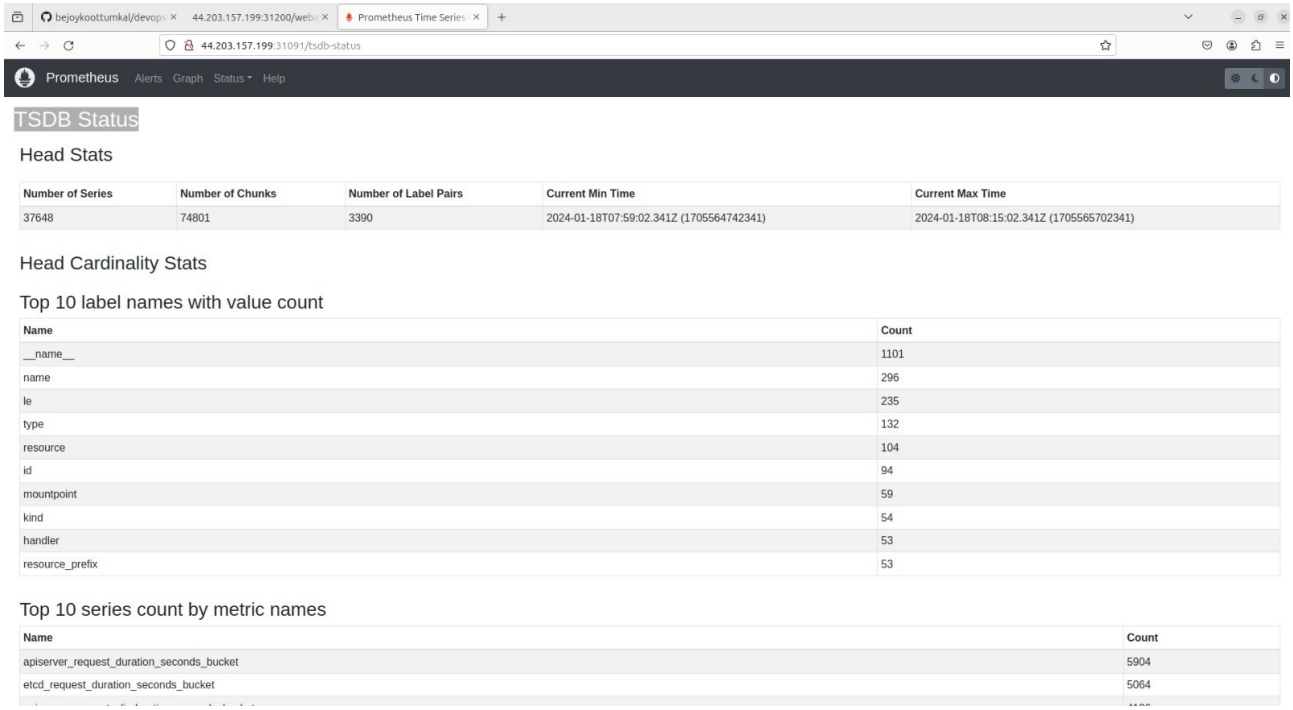
Service Discovery



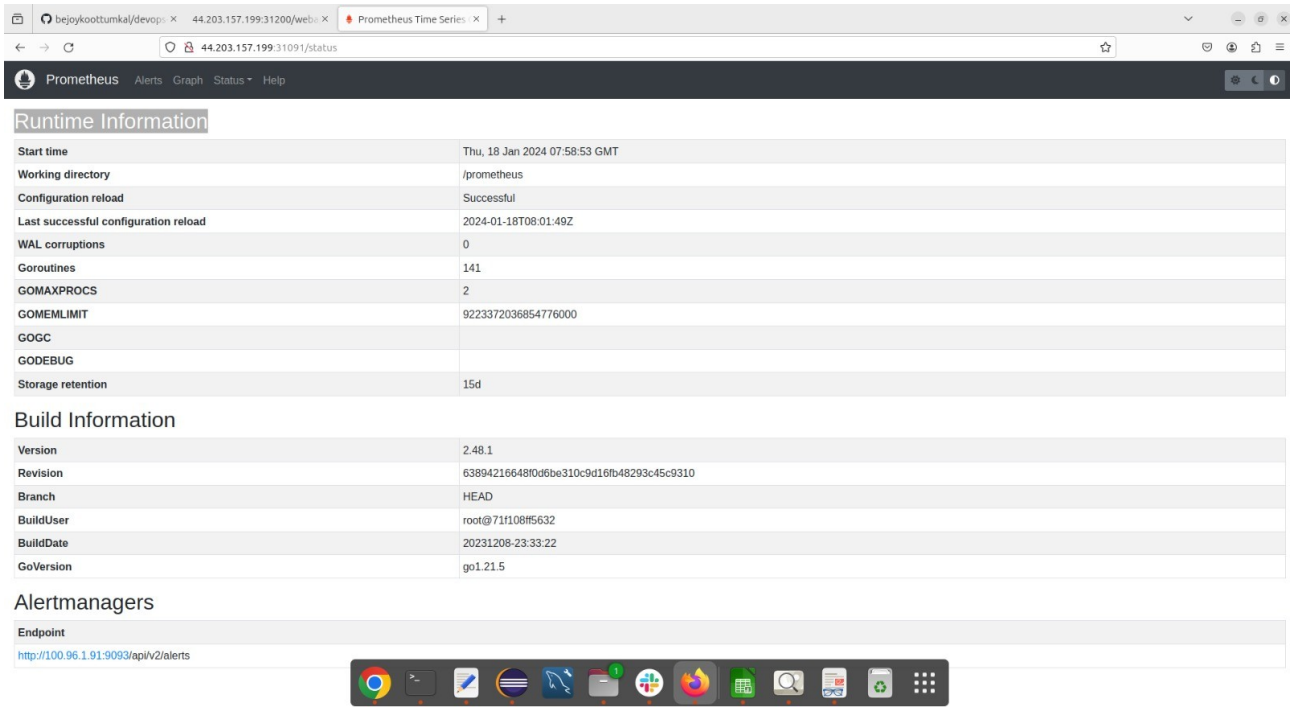
Configuration



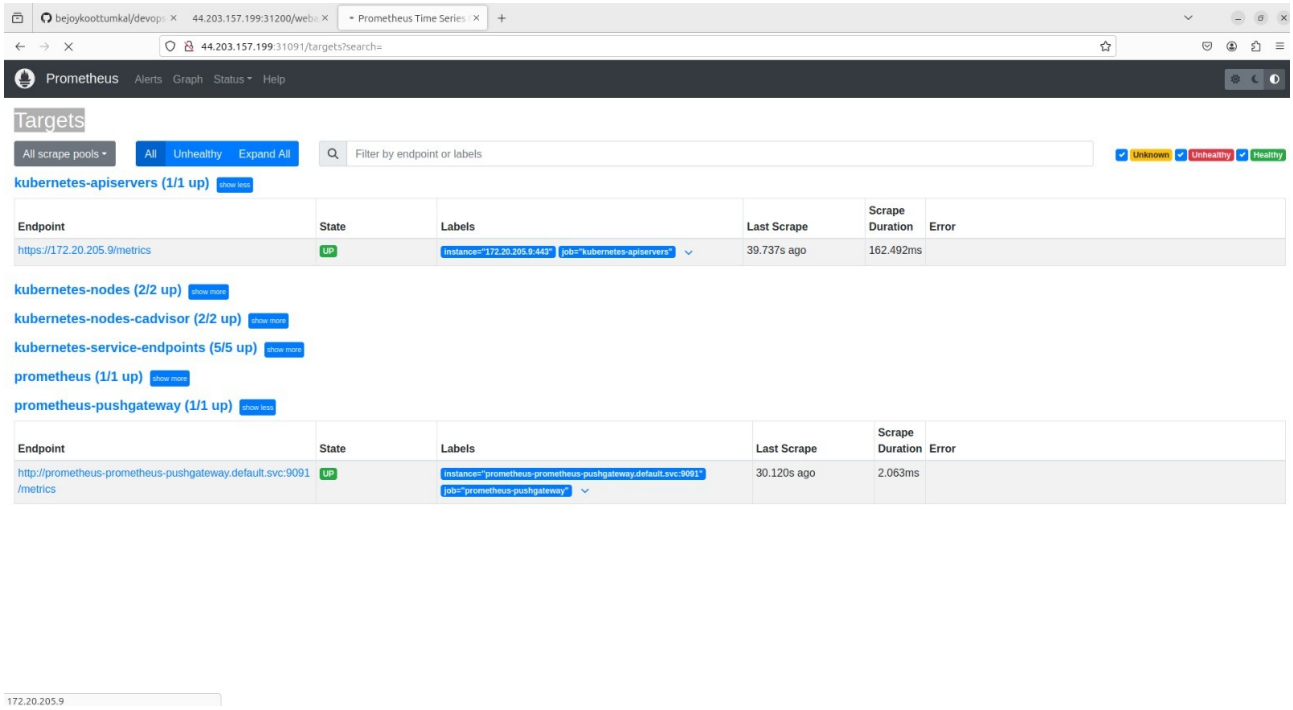
TSDB Status



Runtime Information



Prometheus Targets



Step 4: Monitored the Total Usage, Usage per core, usage breakdown, Memory, and Network on the instance Using the Prometheus

1) Total Usage:

Metric used: node_cpu_seconds_total

Description: This metric represented the total CPU time consumed by the system.

PromQL Query:

sum(rate(node_cpu_seconds_total{mode="user"}[1m])) by (instance)

This PromQL query calculated the per-instance total CPU usage per second over the last 1 minute.

2) Usage per Core:

Metric used: node_cpu_seconds_total

Description: CPU usage was broken down per core by using labels.

PromQL Query:

irate(node_cpu_seconds_total{mode="user"}[1m]) by (instance, cpu)

This query calculated the per-instance and per-core CPU usage per second over the last 1 minute.

3) Usage Breakdown:

Metric used: node_memory_MemTotal, node_memory_MemFree, node_memory_Buffers, node_memory_Cached

Description: A breakdown of memory usage was obtained using various metrics related to memory.

PromQL Query:

node_memory_MemTotal - (node_memory_MemFree + node_memory_Buffers + node_memory_Cached)

This query calculated the used memory by subtracting free memory, buffer, and cache from the total memory.

4) Memory:

Metric used: node_memory_MemTotal, node_memory_MemFree

Description: Total memory and free memory were monitored individually.

PromQL Queries:

node_memory_MemTotal

This query provided the total memory.

node_memory_MemFree

This query provided the amount of free memory.

5) Network:

Metric used: node_network_receive_bytes_total, node_network_transmit_bytes_total

Description: Network usage was monitored.

PromQL Queries:

irate(node_network_receive_bytes_total[1m]) by (instance, device)

This query calculated the per-instance and per-network device incoming data rate over the last 1 minute.

irate(node_network_transmit_bytes_total[1m]) by (instance, device)

This query calculated the per-instance and per-network device outgoing data rate over the last 1 minute.

THANK YOU